中国运筹学会
中国工业与应用数学学会 丛书

应用数学译丛

第3号

最优化方法

[日]茨木俊秀 福岛雅夫著
曾道智译

世界图书出版公司

最 优 化 方 法

〔日〕茨木俊秀　福岛雅夫　著

曾道智　译



世界图书出版公司

北京・广州・上海・西安

1997

本书既考虑到这样的新发展，又注意到通俗易懂地说明了最优化的基本知识到最新成果的内容．也就是，从单纯法开始，精选叙述了

线性规划

网络最优化

组合最优化

非线性规划

的各个领域中有代表性的方法以及最近的话题．这些话题里有上述内点法，也有网络上针对最小成本流的强多项式算法，针对旅行商问题的多面体方法，非线性规划里的逐次二次规划法及接近法等等．另外，还特地用了一章解说最近引人注目的神经网络上的最优化方法．

最后，我们不仅从数学观点上对这些最优化方法进行了理论研究，还强调了它们作为实用工具在现实中被广泛应用的事实．大多数有代表性的最优化算法已有可以方便使用的软件包．但是，有效利用这些成果是以有待解决的问题已被模型化成最优化问题的形式为前提的．这一过程未必简单．在收集客观数据的同时，要求有深刻的洞察力和综合能力．运筹学兼蓄有针对性的知识及技能，但本书完全没涉及到．仅于此指出其重要性，想等别的机会详谈．

尽管仅靠最优化的方法不能解决所有问题，但是我们认为，这方法对解决最优化问题中可以模型化的部分是必不可少的，今后也会继续起到问题解决方法的核心作用．在理解最优化方法方面本书若能有所贡献，即便一点点，我们也很高兴．

茨木俊秀　福岛雅夫

1993 年 6 月　于京都，奈良

图书在版编目(CIP)数据

最优化方法/(日)茨木俊秀,(日)福岛雅夫著;曾道智译.－北京:世界图书出版公司北京公司,1997.3
(应用数学译丛/章祥荪等主编)
ISBN 7－5062－2853－X

Ⅰ.最… Ⅱ.①茨… ②福… ③曾… Ⅲ.最佳化 Ⅳ.O224

中国版本图书馆 CIP 数据核字(97)第 05869 号

茨木俊秀　福島雅夫
最適化の方法
共立出版株式会社,1993

最优化方法
〔日〕茨木俊秀　福岛雅夫著
曾道智　译
责任编辑　高　蓉
世界图书出版公司北京公司出版
北京朝阳门内大街 137 号
北京昌平百善印刷厂印刷
新华书店北京发行所发行　各地新华书店经售

1997 年 4 月第　1　版　　开本:850×1168　1/32
1997 年 4 月第 1 次印刷　　印张:8
印数:0001－2000　　字数:18 万字
ISBN: 7－5062－2853－X/O・175
著作权合同登记　图字:01－96－0117 号
定价:15.00 元

应用数学译丛

中文版前言

中国和日本的的确确是近邻，飞机仅需两个多小时就可以到，我们已经有幸多次访问过中国．北京的故宫、八达岭、西安的兵马俑、曲阜的孔府等等，让我们深深感受到中国四千年历史的源远流长．我们记忆尤新并感到骄傲的是，徒步登上泰山而不是坐缆车．

然而，和这些文化遗产以及美丽大自然比起来，中国年青人奋发图强，刻苦学习的精神面貌给我们留下了更为深刻的印象．这本书能译成中文出版，那怕是能满足一点点这些年青人的求知欲，我们也会因此感到由衷地高兴．

担任翻译工作的曾道智君是位生气勃勃的后起之秀，在京都大学我们研究室里刚获得博士学位．相信他一定能够准确译出专业术语(非常遗憾的是，我们连确认的中文知识都没有)．

茨木俊秀

福岛雅夫

1996 年 4 月　于京都大学

原著前言

如何使用计算机来解决现实中各种各样问题？信息数学积累了这方面的知识并不断深入发展．被称为**最优化** (optimization) 或**数学规划** (mathematical programming) 的领域讨论的是为找出对象问题的最优解决策略而采取的模型化及其方法．其过程是，先把待解决问题用最优化形式描述为在给定的约束条件下找出使某个目标函数达最大 (小) 的解 ", 然后再采用数学上严密的算法来求解．再广泛些，这一领域也被认为是运筹学 (OR) 的一部分．

所谓问题解决方法，除了运筹学和数学规划之外，人们还提出了人工智能 (AI)，专家系统、系统理论，模糊集合，神经元，遗传算法等等各种典型方法，并使它们实用化．这些方法各有各的特征，承担起解决问题某个方面的任务，但决不否定数学规划的作用．这是因为把要解决的问题描述成最优化问题后，没有其它方法能像数学规划那样深刻解析对象问题并给出精密解．

普遍认为，数学规划领域的诞生是在 1947 年，当时 G.B. Dantzig 提出了线性规划的基本算法 — 单纯形算法．至今已有近半个世纪的历史．理论上虽已显成熟，但我们也可以说它还是个正在继续茁壮成长的领域．在这个意义上最近成为热门话题的是始自 1984 年 N. Karmarkar 发表的内点法．这个新方法不仅促进了算法的高速化，还因为这种想法本身获得了美国的专利，成为围绕着知识产权而展开的话题，给人们留下了很深的印象．经内点法刺激的单纯形法也取得了进步，在实用上现在可以解有自数万到数十万个变量的线性规划问题．

第 0 章 绪 论

最优化(optimization) 一般是指在某种状况下作出最好的决策，或者是从几个候选者中选出最好的．这种问题经常用下面的数学模型描述：

> “在给定的 **约束条件**(constraint) 下，找出一个 **决策变量** (decision variable) 的值，使得被称为 **目标函数**(objective function) 的表达愿望尺度的函数达到最小或最大值．”

一般说来决策变量 (以下简称为变量) 有多个，因此用 n 维向量 $x = (x_1, \ldots, x_n)$ 来表示，把问题写成下式[1]．

$$\begin{array}{ll} \text{目标函数：} & f(x) \to \text{最小} \\ \text{约束条件：} & x \in S \end{array} \tag{1}$$

在此，目标函数 f 是定义在包含 S 的适当集合上的实值函数．进一步，S 是该问题变量 x 的可取值之集合，称为问题 (1) 的**可行域** (feasible region)．一般来说，可行域 S 用与变量 x 相关的等式及不等式表示，但也有很多情况下包含有难以用数学式子表达的组合条件．

---

1) 目标函数最大化等价于把它乘以 -1 后最小化，因此本书在没有特地指出的情况下仅处理最小化问题．

满足约束条件 $x \in S$ 的 x 称为问题 (1) 的**可行解** (feasible solution), 满足

$$f(x^*) \leq f(x) \quad (x \in S) \tag{2}$$

的可行解 $x^* \in S$ 称为问题 (1) 的**最优解** (optimal solution). 另外, 在包含可行解 $x^* \in S$ 的适当邻域 $U(x^*)$ 里, 当

$$f(x^*) \leq f(x) \quad (x \in S \cap U(x^*)) \tag{3}$$

成立时, 称 x^* 为问题 (1) 的**局部最优解** (local optimal solution). 不少问题的目标函数或约束条件很复杂, 要找出在全可行域中使目标函数达最小的解非常困难, 这时面临的目标就成为求出局部最优解. 为区别 (2) 的最优解与局部最优解, 有时特地称其为**全局最优解** (global optimal solution).

根据变量, 目标函数, 约束条件的类型, 最优化问题 (1) 可分成几类. 首先, 根据变量, 大致分为变量取连续实数的**连续最优化问题** (continuous optimization problem) 以及取整数或者类似 0, 1 的离散值的**离散最优化问题** (discrete optimization problem). 后者因多用组合性质来表达, 也称为**组合优化问题** (combinatorial optimization problem).

连续最优化问题分为目标函数是线性, 约束条件是线性方程式或(和)不等式组时的**线性规划问题** (linear programming problem) 及目标函数和约束条件未必是线性的**非线性规划问题** (nonlinear programming problem). 后者可进一步分为目标函数是二次, 约束条件是线性的**二次规划问题** (quadratic programming problem), 目标函数是凸函数、约束条件是凸集合[1] 的**凸规划问题** (convex programming problem) 等. 凸规划问题有局部最优

1) 凸集合及凸函数的定义请参看附录 A.1.

解必定是全局最优解的性质，线性规划问题是凸规划问题的特殊情形.

离散最优化问题有各式各样的问题类，把它们具体特征化的主要原因在于其约束条件的结构. 一般说来，有几个变量被限制取整数时的问题称为**整数规划问题** (integer programming problem)，特别是，当变量的值是 0 或 1 的时候称为 **0-1 规划问题** (0-1 programming problem). 另外，约束条件由关联图或网络给出的问题也很多. 例如，处理网络的流的各种**网络流问题** (network flow problem) 及成为困难组合最优化问题典型的**旅行商问题** (traveling salesman problem).

上述线性规划问题，非线性规划问题，整数规划问题等统称为**数学规划问题** (mathematical programming problem).

本书论及各种最优化问题，着重于解说它们的解法 (算法). 虽然通读本书是以大学一二年级程度的数学知识为前提，但是书末的附录综述了常用的有关线性代数，多变量函数，图论及网络的基础知识，请参考.

第 1 章　线性规划：单纯形法

本章介绍解线性规划问题的有代表性的方法——单纯形法之基本思路. 有好几种执行单纯形法的计算方法，这里特别介绍被称为修正单纯形法的方法. 另外，也讲解线性规划里的对偶定理及灵敏度分析.

1.1　线性规划问题

目标函数为一次函数，约束条件为一次等式或不等式所表示的问题称为**线性规划问题** (linear programming problem, LP problem). 讨论线性规划问题的时候，为方便，考虑如下称为**标准形** (standard form) 的特殊形式问题.

$$
\begin{aligned}
&\text{目标函数}: c^T x \longrightarrow \text{最小} \\
&\text{约束条件}: Ax = b,\ x \geq 0
\end{aligned}
\tag{1.1}
$$

这里，变量 x 为 n 维实向量，A 为 $m \times n$ 常数矩阵，b 和 c 分别为 m 维及 n 维常数向量，T 为转置记号. 以下所有的向量全为列向量，同维的两个向量 x, y 所对应的各分量当 $x_i \geq y_i$ 成立时，记为 $x \geq y$.

在一实际应用里出现各种线性规划问题，它们都可以变换成标准形(1.1). 比如，函数 $c^T x$ 的最大化和 $(-c)^T x$ 的最小化等

价，不等式约束条件

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i$$

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i$$

在引进新变量 y_i 后，分别转化为等式约束条件和变量非负条件

$$\sum_{j=1}^{n} a_{ij}x_j + y_i = b_i,\ \ y_i \geq 0 \tag{1.2}$$

$$\sum_{j=1}^{n} a_{ij}x_j - y_i = b_i,\ \ y_i \geq 0 \tag{1.3}$$

另外，不受符号约束的变量 x_j 可像 $x_j = x_j' - x_j''$ 一样表示为两个非负变量 $x_j' \geq 0, x_j'' \geq 0$ 之差，因此，通过适当组合以上步骤，可把任意问题变换为与其等价的标准形．一般把式 (1.2) 的 y_i 称为**松弛变量** (slack variable)，式 (1.3) 的 y_i 称为**剩余变量** (surplus variable)．

1.2 基解和最优解

为讨论方便，以下假定问题 (1.1) 的变量个数 n 比约束条件 (变量非负条件除外) 个数 m 大，约束条件的系数矩阵 A 满足 $\text{rank}\, A = m$[1]．前面的假定在利用松弛变量和剩余变量变换后的标准形里通常成立．后面的假定是说在约束条件里没有 (像能通过组合其它几个条件而得到的) 多余条件式子．

1) 矩阵 A 的列向量中线性无关者的最大个数称为 A 的秩 (rank)，记为 $\text{rank}A$

考虑问题 (1.1) 的等式约束条件

$$Ax = b \tag{1.4}$$

这里，向量 x 的 n 个分量 $x_j\ (j = 1,2,\dots,n)$ 分成 m 个和 $n-m$ 个两组，把它们以适当顺序排列后分别记为 x_B, x_N. 进一步，对应向量 x 的分组，把 $m \times n$ 矩阵 A 也分成 $m \times m$ 矩阵和 $m \times (n-m)$ 矩阵，前面的矩阵记为 B, 后面的记为 N[1] 以下，把这种向量和矩阵划分表示为 $x = [x_B, x_N]$, $A = [B, N]$

对应于某个划分 $x = [x_B, x_N]$, $A = [B, N]$, 式 (1.4) 可写为

$$Bx_B + Nx_N = b, \tag{1.5}$$

因此，如果 B 为非奇异矩阵，通过令 $x_N = 0$, x_B 的值可由式 (1.5) 唯一确定为 $x_B = B^{-1}b$. 这样得到的

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} \tag{1.6}$$

是方程式 (1.4) 的一个特殊解，称为问题 (1.1) 的 **基解** (basic solution). 特别是，在式 (1.6) 里 $x_B = B^{-1}b \geq 0$ 成立时，该基解满足问题 (1.1) 的约束条件，因此称为 **基可行解** (basic feasible solution). 一般来说，问题 (1.1) 的可行解有无数个，而基可行解仅有有限个[2] 对应于式 (1.6) 的基解，称 B 为 **基矩阵** (basic matrix), 向量 x_B 的各分量称为 **基变量** (basic variable), N 称为

---

1) 在 $n = 5, m = 3$, $x = (x_1, \dots, x_5), A = (a_1, \dots, a_5)$ 时，如果 $x_B = (x_2, x_5, x_1), x_N = (x_3, x_4)$ 则 $B = (a_2, a_5, a_1)$, $N = (a_3, a_4)$. 这里，a_i 为矩阵 A 的第 i 列向量.

2) 因为可能的划分 $x = [x_B, x_N]$ 最多有 $\begin{pmatrix} n \\ m \end{pmatrix} = \frac{n!}{m!(n-m)!}$ 种，基可行解的个数不可能超过这个数

非基矩阵 (nonbasic matrix), 向量 x_N 的各分量称为 **非基变量** (nonbasic variable).

作为具体的例子，考虑如下的约束条件.

$$
\begin{aligned}
&2x_1 + x_2 \leq 6 \\
&x_1 + 2x_2 \leq 6 \\
&x_1 \geq 0,\ \ x_2 \geq 0
\end{aligned}
\tag{1.7}
$$

因它不是标准形，用前节所述的方法变换成如下的标准形.

$$
\begin{aligned}
&2x_1 + x_2 + x_3 = 6 \\
&x_1 + 2x_2 + x_4 = 6 \\
&x_i \geq 0 \qquad (i = 1, 2, 3, 4)
\end{aligned}
\tag{1.8}
$$

这里， x_3, x_4 是松弛变量. 式 (1.8) 里有下面 (a)–(f) 的六个基解. 这些基解里面，除 (b) 和 (e) 以外，其余四个为基可行解.

(a) $x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$

$x_N = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

(b) $x_B = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ -6 \end{bmatrix}$

$x_N = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

(c) $x_B = \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$

$x_N = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$$(d)\ x_B = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$x_N = \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$(e)\ x_B = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ -6 \end{bmatrix}$$

$$x_N = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$(f)\ x_B = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

$$x_N = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

式 (1.8) 里变量的维数是 4, 无法直接用图表示, 但是与式 (1.8) 等价的原不等式组 (1.7) 在 (x_1, x_2) 平面上可表示为图 1.1. 图 1.1 中, 注意包括纵轴和横轴四条直线分别表示 $x_j = 0$ $(j = 1,2,3,4)$. 于是, 这四条直线中每两条之交点分别对应于上述六个基解, 特别是, 表示可行域四边形的顶点成为基可行解. 该例中可行域为四边形, 一般情况下, 线性规划问题可行域为 n 维欧几里得空间 R^n 的凸多面体, 基可行解对应于该凸多面体的顶点 [1].

---

1) 空间 R^n 里, 通过适当移动 $(n-1)$ 维子空间得到的集合称为超平面 (hyperplane). 三维空间 R^3 内的平面以及二维空间 R^2 内的直线都是 (各自空间里的) 超平面. 超平面可用某个向量 $a \neq 0$ 及实数 b 表示成 $\{x \in R^n \mid a^T x = b\}$, 进一步, 一个超平面决定两个半空间 $\{x \in R^n \mid a^T x \geq b\}$ 和 $\{x \in R^n \mid a^T x \leq b\}$. 几个半空间的共同部分称为凸多面体 (polyhedral convex set).

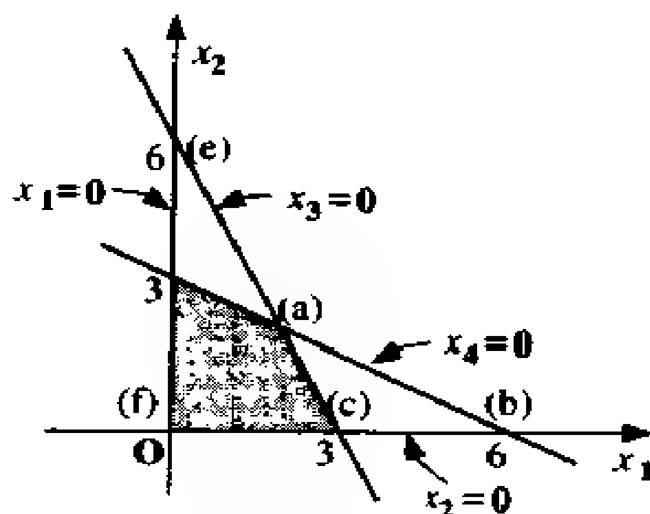
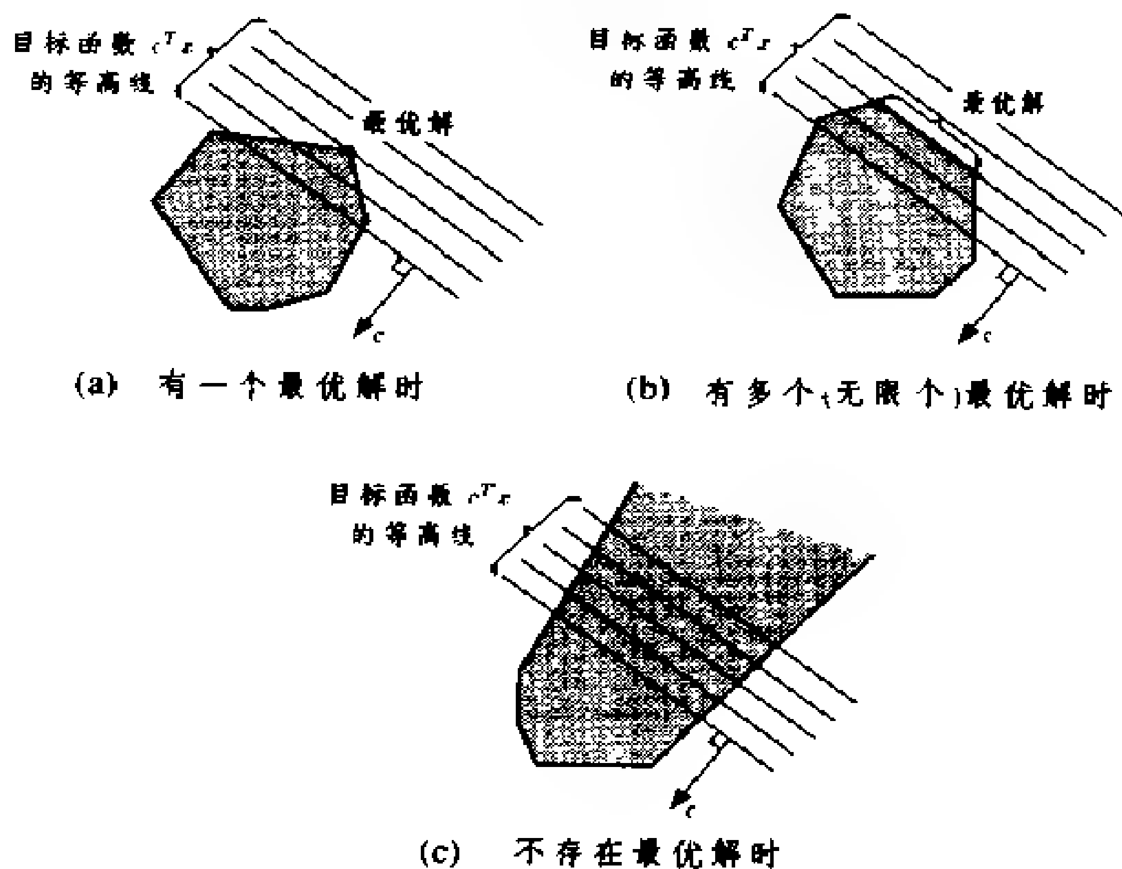


图 1.1 基底解



(a) 有一个最优解时

(b) 有多个(无限个)最优解时

(c) 不存在最优解时

图 1.2 线性规划问题的最优解

在空间 R^n 中作为目标函数的一次函数 c^Tx 的等高线是平

行超平面，因此线性规划问题的最优解的集合，要么像图 1.2 (a) 一样由一个顶点构成，要么像图 1.2 (b) 一样由可行域 (凸多面体) 的一个面构成，或者像图 1.2 (c) 那样可行域内目标函数值可以任意小．特别是，当问题有最优解时，可行域的顶点之中一定存在有成为最优解者．把以上各项综合一下就得到下面的定理．

定理 1.1 (线性规划问题的最优解的特征) 线性规划问题若有最优解，则基可行解中一定存在有最优解．

根据这个定理知道，为了找出线性规划问题的最优解没必要考虑 (无限个) 可行解的全体，可限定以具有式 (1.6) 的形式的 (有限个) 基可行解为对象进行搜索．基于这个想法进行有效的最优基解搜索的是 1947 年 G.B. Dantzig 所设计的**单纯形法** (simplex method).

1.3 单纯形法

单纯形法是逐步生成有较小目标函数值的基可行解，最终到达最优解的迭代法．这里，首先假定对标准形问题 (1.1) 得到了一个给出基可行解的划分 $x = [x_B, x_N]$, $A = [B, N]$ (求基可行解的一般方法在 1.6 节叙述). 此时，根据式 (1.5), 因为

$$x_B = B^{-1}b - B^{-1}Nx_N \tag{1.9}$$

目标函数的系数向量也同样划分为 $c = [c_B, c_N]$, 若用 z 表示目标函数值，则 z 可用仅含有非基变量 x_N 的形式表示如下[1].

---

1) 式 (1.10) 的 $j \in N$ 表达式里 N 被看成是非基变量的下标集合．以下也同样，N 不仅表示非基矩阵，也用来表示非基变量下标集合．

$$\begin{aligned} z &= c_B^T x_B + c_N^T x_N \\ &= c_B^T B^{-1} b + (c_N^T - c_B^T B^{-1} N) x_N \\ &= z_0 + \sum_{j \in N} (c_j - z_j) x_j \end{aligned} \tag{1.10}$$

这里，z_0 和 z_j 是用

$$z_0 = c_B^T B^{-1} b \tag{1.11}$$

$$z_j = c_B^T B^{-1} a_j \tag{1.12}$$

定义的数，a_j 表示矩阵 A 的第 j 列向量. 特别是，式 (1.11) 中 z_0 表示基可行解 $x = [x_B, x_N] = [B^{-1}b, 0]$ 对应的目标函数值. 另外，式 (1.10) 里，$c_j - z_j$ 表示对应于非基变量 x_j 的变化，目标函数值的变化率，称为非基变量 x_j 的**相对成本系数** (relative cost coefficient).

现在，假定对应于某个划分 $x = [x_B, x_N]$, $A = [B, N]$, 由式 (1.12) 所决定的 z_j 满足

$$c_j - z_j \geq 0 \qquad (j \in N) \tag{1.13}$$

这时，因为任意可行解 x 满足 $x_j \geq 0 (j \in N)$, 记它的目标函数值为 z, 则由式 (1.11), 成立有

$$z = z_0 + \sum_{j \in N} (c_j - z_j) x_j \geq z_0 \tag{1.14}$$

也就是，因为基可行解 $x = [x_B, x_N] = [B^{-1}b, 0]$ 的目标函数值不比任何一个可行解的目标函数值大，由此知道它即为问题 (1.1) 的最优解.

反之，式 (1.13) 不成立时，存在有满足

$$c_p - z_p < 0 \tag{1.15}$$

的 $p \in N$. 因此，把 x_p 以外的非基变量固定为现在的值 0, 将 x_p 值由 0 增至 $\Delta(\geq 0)$, 根据式 (1.11) 和 (1.15), 目标函数值 z 可由 z_0 减至 $z_0 + (c_p - z_p)\Delta \ (\leq z_0)$.

这时，为了保持问题 (1.1) 的可行性，有必要让基变量 x_B 在满足式 (1.9) 的条件下变化，故做如下修正. 式 (1.9) 可写成

$$x_B = B^{-1}b - B^{-1}\sum_{j \in N} a_j x_j$$

若取 $x_p = \Delta, x_j = 0 \ (j \in N, j \neq p)$, 则 x_B 的值成为

$$x_B = B^{-1}b - B^{-1}a_p\Delta \tag{1.16}$$

这里，为表达方便，令

$$\bar{b} = B^{-1}b$$

$$y_p = B^{-1}a_p$$

把向量 $x_B, \bar{b}, y_p$ 的分量分别记为 $x_B = (x_{B_1}, x_{B_2}, \ldots, x_{B_m})^T$, $\bar{b} = (\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_m)^T$, $y_p = (y_{1p}, y_{2p}, \ldots, y_{mp})^T$, 则式 (1.16) 可写成下式[1].

$$\begin{bmatrix} x_{B_1} \\ x_{B_2} \\ \vdots \\ x_{B_m} \end{bmatrix} = \begin{bmatrix} \bar{b}_1 \\ \bar{b}_2 \\ \vdots \\ \bar{b}_m \end{bmatrix} - \begin{bmatrix} y_{1p} \\ y_{2p} \\ \vdots \\ y_{mp} \end{bmatrix} \Delta \tag{1.17}$$

1) 考虑到划分 $x = [x_B \ x_N]$ 里基变量 x_B 的分量排列顺序，把第 i 个变量表示为 x_{B_i}. 比如，若 $m = 3$, $x_B = (x_2, x_5, x_1)^T$, 则 $x_{B_1} = x_2, x_{B_2} = x_5, x_{B_3} = x_1$.

根据问题(1.1)的约束条件，变量必须全部非负，在式(1.17)右边不为负的范围里取最大的Δ，即以

$$\Delta = \frac{b_r}{y_{rp}} = \min\left\{ \frac{b_i}{y_{ip}} \,\middle|\, y_{ip} > 0 \ (i = 1, 2, \dots, m) \right\} \tag{1.18}$$

作为变量x_p的新值．其中，式(1.18)意味着对成立$y_{ip} > 0$的i取最小．将达到最小的下标i记为r．这样定下的

$$x_{B_i} = b_i - y_{ip}\Delta \qquad (i = 1, 2, \dots, m) \tag{1.19}$$

$$x_p = \Delta \tag{1.20}$$

$$x_j = 0 \qquad (j \in N,\ j \neq p) \tag{1.21}$$

是问题(1.1)的可行解，特别是关于变量x_{B_r}成立

$$x_{B_r} = 0 \tag{1.22}$$

根据式(1.19)-(1.22)，x_{B_r}和$x_j (j \in N, j \neq p)$合起来$n - m$个变量的取值为0，取值为正的只能是除此之外的m个变量$x_{B_1}, \dots, x_{B_{r-1}}, x_p, x_{B_{r+1}}, \dots, x_{B_m}$．因此，把这$m$个变量看成新的基变量，其他$n - m$个变量看成非基变量，就得到新的划分．进一步，可以证明以向量$a_{B_1}, \dots, a_{B_{r-1}}, a_p, a_{B_{r+1}}, \dots, a_{B_m}$为列的$m \times m$矩阵是非奇异的，因此这个新的划分具有前节所讲的基矩阵的资格．

以上的操作通过替换基变量x_{B_r}和非基变量x_p来确定新的基解，称之为**转轴运算**(pivoting)．另外，称这时的x_{B_r}为出基变量，x_p为入基变量．这样一来得到新的基可行解，重复上述步骤利用式(1.11)，(1.12)进行计算，并利用(1.13)进行最优判定．

若式(1.17)中不存在有使$y_{ip} > 0$的i，则不论Δ有多大，基变量决不可能为负．这意味着在保持问题(1.1)的可行性的同

时，目标函数值可以任意小，也就是问题 (1.1) **无界** (unbounded)，因此可以结束计算.

综合以上计算步骤得到如下算法.

算法 SIMPLEX (单纯形法)

第一步 (初始化): 确定给出基可行解的划分 $A = [B, N]$，计算 $x_B = \bar{b} := B^{-1}b$

第二步 (最优判定): 计算向量 $w = (B^T)^{-1}c_B$，对所有的非基分量 $j \in N$ 求出 $z_j = w^T a_j$. 如果 $c_j - z_j \geq 0 \, (j \in N)$，则现在的基解 $[x_B, x_N] = [\bar{b}, 0]$ 是最优解，因而结束计算. 否则选取一个满足 $c_p - z_p < 0$ 的 $p \in N$ 进入第三步.

第三步 (转轴运算): 计算向量 $y_p = B^{-1}a_p$. 如果 $y_p \leq 0$，则问题 (1.1) 没有有界解，因而结束计算. 否则，找出成立

$$\Delta := \bar{b}_r / y_{rp} = \min \left\{ \bar{b}_i / y_{ip} \mid y_{ip} > 0 \, (i = 1, 2, \ldots, m) \right\}$$

的 r. 把基矩阵 B 的列向量 a_{B_r} 与 a_p 交换，确定新的基矩阵 B. 进一步，记新基变量的值为 $x_{B_i} = \bar{b}_i - y_{ip}\Delta \, (i = 1, 2, \ldots, m,\ i \neq r)$ 和 $x_p = \Delta$. 令基变量的下标集合和非基变量的下标集合分别为 $B := B \cup \{p\} - \{B_r\}, N := N \cup \{B_r\} - \{p\}$ 回到第二步.

执行算法 SIMPLEX 时，有必要事先找出一个基可行解，对一般问题来说这并不一定是件容易的事情. 关于这一点要在 1.6 节详细说明.

第二步里用向量

$$w = (B^T)^{-1}c_B$$

为媒介，计算各非基变量对应的相对成本系数 $c_j - z_j$. 向量 w 的分量一般称为**单纯形乘子** (simplex multipliers). 另外，入基变量 x_p 从成立 $c_p - z_p < 0$ 之中适当选取任意一个都行，特别注意到相对成本系数 $c_j - z_j$ 表示增加 x_j 时目标函数的变化率，为减少算法的迭代次数，选取成立

$$c_p - z_p = \min_{j \in N} \{c_j - z_j\}$$

的 x_p 一般被认为是有效的[1].

在第三步里确定出基变量 x_{B_r} 时，使比值 $\bar{b}_i / y_{ip}$ 达最小的 i 有可能存在多个，出现无法唯一确定 r 的情况. 实际计算时从中适当选取任意一个都行，但是像下节要讲的那样，在从理论上讨论单纯形法的有限收敛性时，如何决定入基变量和出基变量成为重要的问题.

下面举例用单纯形法解如下问题.

$$\begin{aligned} &\text{目标函数：} -3x_1 - 2x_2 \longrightarrow \text{最小} \\ &\text{约束条件：} 2x_1 + x_2 + x_3 \qquad = 6 \\ &\qquad\qquad\quad x_1 + 2x_2 \qquad + x_4 = 6 \\ &\qquad\qquad\quad x_i \geq 0 \ (i = 1, \ldots, 4) \end{aligned} \tag{1.23}$$

初始解

第一步：取 $B = [a_3, a_4]$ 为初始基矩阵. $x_B = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$ (目标函数值 $z_0 = 0$).

迭代 1

1) 变量数目非常多时，严密找出最小的 $c_j - z_j$ 本身需要相当的计算量. 这种情况下，提高算法全体的计算效率的有效方法是把变量分成几个组来处理.

第二步：$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $z_1 = [0,0]\begin{bmatrix} 2 \\ 1 \end{bmatrix} = 0$, $z_2 = [0,0]\begin{bmatrix} 1 \\ 2 \end{bmatrix} = 0$. 因有 $c_1 - z_1 = -3 < 0, c_2 - z_2 = -2 < 0$ 选取变量 x_1, x_2 中任意一个入基，目标函数值都减少. 在此取 $x_1(p = 1)$ 为入基变量进入第三步

第三步：$y_1 = \begin{bmatrix} y_{11} \\ y_{21} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$,

因为 $\Delta = \min\{6/2, 6/1\} = 3$ 故 $r = 1$, 也就是 $x_{B_1}(= x_3)$ 要出基. 更新为 $B = [a_1, a_4]$ 新基变量成为 $x_B = \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} \Delta \\ 6 - \Delta \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$ (目标函数值 $z_0 = c_B^T x_B = [-3, 0]\begin{bmatrix} 3 \\ 3 \end{bmatrix} = -9$). 回到第二步.

迭代 2

第二步：$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 0 \end{bmatrix} = \begin{bmatrix} -3/2 \\ 0 \end{bmatrix}$,

$z_2 = [-3/2, 0]\begin{bmatrix} 1 \\ 2 \end{bmatrix} = -3/2$, $z_3 = [-3/2, 0]\begin{bmatrix} 1 \\ 0 \end{bmatrix} = -3/2$ 因为 $c_2 - z_2 = -2 - (-3/2) = -1/2 < 0, c_3 - z_3 = -(-3/2) = 3/2 > 0$, 只有变量 x_2 具有入基资格. 令 $p = 2$ 进入第三步

第三步 $y_2 = \begin{bmatrix} y_{12} \\ y_{22} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 3/2 \end{bmatrix}$,

因为 $\Delta = \min\{3/(1/2), 3/(3/2)\} = 2$ 故 $r = 2$, 也就是 $x_{B_2}(= x_4)$ 要出基. 更新为 $B = [a_1, a_2]$ 新基变量成为 $x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 - (1/2)\Delta \\ \Delta \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$ (目标函数值 $z_0 = c_B^T x_B = [-3, -2]\begin{bmatrix} 2 \\ 2 \end{bmatrix} = -10$) 回到第二步.

迭代 3

第二步：$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ -2 \end{bmatrix} = \begin{bmatrix} -4/3 \\ -1/3 \end{bmatrix}$,

$z_3 = [\ 4/3, -1/3]\begin{bmatrix}1\\0\end{bmatrix} = -4/3, z_4 = [\ 4/3,\ \ 1/3]\begin{bmatrix}0\\1\end{bmatrix} = -1/3$ 因为 $c_3 - z_3 = (-4/3) - 4/3 > 0, c_4 - z_4 = (\ 1/3) - 1/3 > 0$, 故现在的基解 $x = (2,2,0,0)^T$ 是最优的.

上面所述的单纯形法一般称为**修正单纯形法** (revised simplex method). 这个方法的特征在于, 各次迭代都是利用当时的基矩阵和所给问题的数据来计算 w 和 y_p 为使算法能有效地执行, 有必要在计算上进一步下工夫. 这样的方法将在 1.5 节说明.

除修正单纯形法外, 同样具有代表性的单纯形法的计算方式还有利用表格的、称为**单纯形列表法** (simplex tableau) 的方式. 单纯形表是基于某个基的划分来表达问题的工具, 对应每个基解都存在有一个单纯形表. 单纯形表里包含有执行单纯形法而需要的所有信息, 这种方式没必要像修正单纯形法那样重新计算 w 和 y_p, 付出的代价是每次迭代都要全部重写表, 因而必须进行修正单纯形法所不必要的计算. 利用表的单纯形法和修正单纯形法之中, 哪个能在较少的计算时间内结束依赖于待解问题的形状及规模. 对大规模问题有可能利用问题的结构使计算有效化, 因此线性规划算法的商用软件里一般都采用修正单纯形法.

1.4 理论收敛性

如前节所述, 经过转轴运算, 对应于 $c_p - z_p < 0$ 的非基变量 x_p 进入基, 得到新的基可行解时, 目标函数值仅仅减少 $|c_p - z_p|\Delta$. 这里, Δ 是式 (1.18) 所定义的量. 因为 $\bar{b} = B^{-1}b \geq 0$, 根据式 (1.18) 恒有 $\Delta \geq 0$, 特别是当

$$\bar{b} > 0 \tag{1.24}$$

成立时[1] 有 $\Delta > 0$, 目标函数值严格减少. 另外, 对应各基可行解目标函数的值是唯一决定的, 单纯形法的各次迭代里若恒成立式 (1.24), 则逐次生成的基可行解之中应该不存在有相同的. 因为基可行解数目是有限的, 在这种情况下, 迭代不会无限次重复下去, 而在有限次迭代后结束计算. 基可行解 $[x_B, x_N] = [\bar{b}, 0]$ 满足式 (1.24) 时, 称它为**非退化**(nondegenerate) 的. 反之, 存在 $\bar{b}_i = 0$ 的 i 时, 称该基解为**退化**(degenerate) 的. 以上的讨论可综合成如下定理.

定理 1.2 (单纯形法的有限收敛性 — 非退化的情况) 若所给定的线性规划问题可以求出初始基可行解, 而且计算中出现的所有基可行解都是非退化的, 则利用单纯形法在有限次迭代后, 要么找到最优解, 要么识别出问题无有界解从而结束计算.

该定理本质上依赖于对基解的非退化性假定. 为观察退化基解究竟表达怎样一种现象, 考虑如下问题.

$$
\begin{aligned}
\text{目标函数:}\ & -3x_1 - 2x_2 \longrightarrow \text{最小} \\
\text{约束条件:}\ & 2x_1 + x_2 + x_3 \qquad\quad = 6 \\
& x_1 + 2x_2 \qquad + x_4 \quad = 6 \\
& x_1 \qquad\qquad\qquad + x_5 = 3 \\
& x_i \geq 0 \quad (i = 1, 2, \ldots, 5)
\end{aligned}
\tag{1.25}
$$

该问题里三种基的划分 $x_B = (x_1, x_2, x_4), x_B = (x_1, x_3, x_4), x_B = (x_1, x_4, x_5)$ 所确定的基可行解都是退化的, 进一步容易验证它们对应于同一个可行解 $(x_1, x_2, x_3, x_4, x_5) = (3, 0, 0, 3, 0)$ 另外,

1) 式 (1.24) 表示 $\bar{b}$ 的所有分量为正.

在 (x_1, x_2)- 平面上表示问题 (1.25) 可行域的图 1.3 里，该基可行解对应于 $x_2 = 0, 2x_1 + x_2 = 6$ ($x_3 = 0$) 及 $x_1 = 3$ ($x_5 = 0$) 三条直线的相交点. 平面上随机放上三条直线，它们刚好交于一点的情况是比较少的，因此，可以认为类似该例的退化基解是相当特殊的情况. 但是现实问题里 (该问题的结构上) 退化基解出现的情况决不是稀奇的.

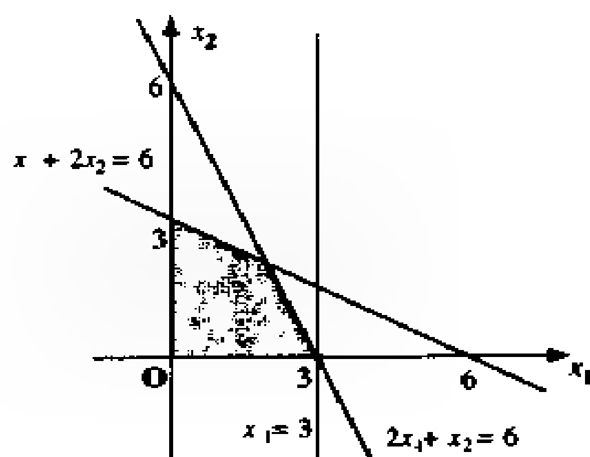


图 1.3　退化基解

单纯形法的某迭代步里出现退化基可行解时，式 (1.18) 的 Δ 可能成为 0. 因此若进行转轴运算，虽然基变量和非基变量的组合要变，但是解 x 实质上不动，目标函数的值也不变化. 另外，这个时候，在入基变量无法唯一确定，而且出基变量也无法唯一确定的情况下，根据其选法，几次转轴运算之后回到和开始时一样的基变量与非基变量的组合之可能性也是有的. 这种现象称为**循环**(cycling). 如果出现循环，单纯形法将无数次重复一样的计算，决不可能到达最优解.

这样一考虑可能就认为单纯形法欠缺可靠性，但是实际上不必过分担心循环. 因为，即使基解的退化是产生循环的必要条件也不是充分条件 [1]. (尽管线性规划法的教科书里出现产生

1) 请对问题 (1.25) 用单纯形法试算一下. 大概经过退化基解将到达最优解

循环的人工例子）现实问题里几乎没有碰到循环现象的报告，实用上忽视循环的可能性也无关紧要.

然而，理论上，怎样才能使单纯形法决不产生循环是很重要的问题. 作为这么一类方法，从 1950 年代开始就出现了**扰动法** (perturbation method) 和**字典法** (lexicographic method)，但是在 1977 年提出的称为**最小下标规则** (smallest subscript rule; 也称 **Bland 规则**) 之思想非常简单.

定理 1.3 (单纯形法的有限收敛性——最小下标规则) 各迭代中，入基变量或出基变量存在多个候补的情况下，恒选取具有最小下标 j 的变量 x_j. 这时单纯形法在有限次迭代后结束计算.

最小下标规则作为防止循环的方法非常有魅力，但实际上，比如前节所述的那样，和选取使 $c_j - z_j$ 最小的变量 x_j 入基的方法相比有需要较多次迭代的倾向，因此不能照搬原样使用.

根据以上讨论，我们知道了单纯形法具有有限收敛性. 那么，它的迭代次数要多大呢？根据至今长年积累的经验来看，对线性规划问题 (1.1) 应用单纯形法时，迭代次数主要依赖于约束条件的数目 m，一般认为在 $1.5m$ 到 $3m$ 之间. 这意味着即使问题的规模增大，所导致的计算量的增大是比较少的. 这是实用上相当令人满意的性质.

但是上面关于迭代次数的估计到底仅仅是经验性的东西，并不是理论上的保证. 实际上， V. Klee 和 G.J. Minty 对某个特殊形状的问题用单纯形法时，证明了到达最优解需要 $2^m - 1$ 次迭代. 也就是说，用计算复杂性理论的术语，单纯形法不是多项式时间算法，而是在最坏的情况下需要指数时间计算量的

算法.

这样就产生了是否存在解线性规划问题，即使在最坏情况下也可在多项式时间内解出的算法，换句话说，线性规划问题是否属于P类的问题. 1979年L.G. Khachiyan第一次对该问题给出了肯定回答. Khachiyan所考察的被称为**椭球法** (ellipsoid method) 的方法虽然在理论上是多项式时间算法，可是它的实际计算时间比起单纯形法显然要长，因此还没有达到实用化的地步. 然而1984年N. Karmarkar发表了新的多项式时间算法，它在实际计算效率方面也被认为比单纯形法要强. 他的方法现在一般被称为**内点法** (interior method)，渐渐地被评价为代替单纯形法的强有力方法. 本书在第5章里要对这些方法进行讲解.

1.5 单纯形法的有效化

1.3节所述的单纯形法算法中，需要计算量最多的是计算初始解 $x_B = B^{-1}b$ 以及各迭代中向量 $w = (B^T)^{-1}c_B$, $y_p = B^{-1}a_p$. 为求出它们只要分别把 x_B, w, y_p 作为变量，解下面三个一次方程式

$$Bx_B = b \tag{1.26}$$

$$B^T w = c_B \tag{1.27}$$

$$By_p = a_p \tag{1.28}$$

但是，基矩阵 B 在每次迭代中变化，每次从头解方程式 (1.27) 和 (1.28) 的计算量相当大. 因此，注意到当前的基矩阵 B 除了新入基的变量对应的一列后与前次迭代的基矩阵相同这个事实，考虑如何有效利用到前次迭代为止得到的信息来求解.

首先，考虑有关初始基可行解的计算. 实际上初始基矩阵为单位矩阵的情况不少 (参照 1.6 节). 但是因为这里同时也考虑后继基矩阵的再分解，故设定一般的情况. 令初始基矩阵为 $B^{(0)}$，因 $B^{(0)}$ 是非奇异的，可以用某个下三角矩阵 L 和上三角矩阵 U 表示为 [1]

$$B^{(0)}=LU \tag{1.29}$$

式 (1.29) 的表现形式称为矩阵 $B^{(0)}$ 的 **LU 分解** (LU decomposition)，其实际计算方法在一般的数值分析教科书里都有. 于是，这里省略计算法的详细细节，仅仅述说一个注意事项. 一般，问题的规模变大时，系数矩阵 A 中非零分量比起零分量来说会变少而有成为**稀疏矩阵** (sparse matrix) 的趋势. 矩阵为稀疏的时候，让计算机仅记忆非零分量的有关信息便可以有效利用记忆空间，用上这个技巧后计算也得以加速. 另外，可以认为，系数矩阵 A 为稀疏时的问题里基矩阵 B 也是稀疏的. 对这种问题，人们下工夫研究了 LU 分解的计算方法，在把矩阵 B 的行和列作适当的置换后，使矩阵 L 和 U 成为稀疏的而且使得计算误差的影响很小.

在得到矩阵 $B^{(0)}$ 的 LU 分解后，方程式 (1.26) 可表示为

$$LUx_B=b \tag{1.30}$$

这可分成两步来解. 首先解以 z 为未知数的方程式

$$Lz=b \tag{1.31}$$

以得到的解 z 为右边的常数，有方程式

$$Ux_B=z \tag{1.32}$$

再求出这个方程的解 x_B. 注意到这些方程式的系数矩阵的形状、方程式 (1.31) 里从变量 z 的第一个分量开始依顺方向 (前

1) 正方矩阵 $L=(l_{ij})$ 为下三角的是指如果 $i<j$ 则成立 $l_{ij}=0$，正方矩阵 $U=(u_{ij})$ 为上三角的是指如果 $i>j$ 则成立 $u_{ij}=0$

进)代入、方程式 (1.32) 里从变量 x_B 的最后一个分量开始依反方向 (后退) 代入，可以一个接一个地确定其值. 于是，一旦计算了 LU 分解 (1.29), 就容易求出方程式 (1.30) 的解.

下一步，考虑方程式 (1.27) 和 (1.28) 把第 k 次迭代结束时得到的基矩阵记为 $B^{(k)}$. 现假定在第 k 次迭代里、基变量 x_{B_r} 出基，取而代之的是非基变量 x_p 入基，矩阵 $B^{(k)}$ 是用向量 a_p 替代前一步的基矩阵 $B^{(k-1)}$ 的第 r 列 a_{B_r} 后的矩阵，根据 y_p 的定义 $y_p = B^{-1}a_p$, 成立有

$$
\begin{aligned}
B^{(k)} &= B^{(k-1)} + (a_p - a_{B_r})e_r^T \\
&= B^{(k-1)}(I + (y_p - e_r)e_r^T) \\
&= B^{(k-1)}E^{(k)}
\end{aligned}
\tag{1.33}
$$

这里，e_r 是第 r 分量为 1 其余全为 0 的单位向量，$E^{(k)}$ 为单位矩阵的第 r 列被向量 y_p 替代后的矩阵 [1)]

$$
E^{(k)} = \begin{bmatrix}
1 & & & y_{1p} & & & \\
& \ddots & & \vdots & & & \\
& & 1 & \vdots & & & \\
& & & y_{rp} & & & \\
& & & \vdots & 1 & & \\
& & & \vdots & & \ddots & \\
& & & y_{mp} & & & 1
\end{bmatrix}
\tag{1.34}
$$

于是，根据式 (1.29), (1.33), 第 k 次迭代结束时的基矩阵 $B^{(k)}$ 可表示如下.

$$
B^{(k)} = LUE^{(1)}E^{(2)}\cdots E^{(k)}
\tag{1.35}
$$

1) 用计算机处理式 (1.34) 的矩阵 $E^{(k)}$ 时，只要记忆向量 y_p 的值以及它将要替代的是 $E^{(k)}$ 的第 r 列这个事实就够了.

称它为基矩阵的积形式 (product form)[1].

若利用积形式 (1.35), 式 (1.27), (1.28) 可分别写成

$$E^{(k)T}\left(E^{(k-1)T}\left(\cdots\left(E^{(1)T}\left(U^T\left(L^Tw\right)\right)\right)\cdots\right)\right)=c_B \quad (1.36)$$

$$L\left(U\left(E^{(1)}\left(\cdots\left(E^{(k-1)}\left(E^{(k)}y_p\right)\right)\cdots\right)\right)\right)=a_p \quad (1.37)$$

故式 (1.27) 的解 w 可由式 (1.36) 从外侧括弧开始依次分解得到一连串的方程式

$$\begin{aligned}&E^{(k)T}u_1=c_B, E^{(k-1)T}u_2=u_1, \cdots, E^{(1)T}u_k=u_{k-1},\\&U^Tu_{k+1}=u_k, L^Tw=u_{k+1}\end{aligned} \quad (1.38)$$

逐次计算其解 $u_1,\cdots,u_{k+1},w$ 而求得. 同样式 (1.28) 的解 y_p 可由式 (1.37) 所构成的一连串方程式

$$\begin{aligned}&Lv_1=a_p, \quad Uv_2=v_1, \quad E^{(1)}v_3=v_2, \cdots,\\&E^{(k-1)}v_{k+1}=v_k, E^{(k)}y_p=v_{k+1}\end{aligned} \quad (1.39)$$

逐次计算其解 $v_1,\dots,v_{k+1},y_p$ 而得到.

这里, 注意包含三角矩阵 L 和 U 的方程式可像刚才所述的那样用前进代入或者后退代入法容易解出,包含矩阵 $E^{(1)},\dots,E^{(k)}$ 的方程式在考虑这些矩阵的特殊形状 (1.34) 后可以非常简单地解出. 于是, 如果保持基矩阵的积形式 (1.35), 方程式 (1.27) 和 (1.28) 的解 w、y_p 的计算就很容易进行. 另外, 各次迭代里基矩阵的积形式 (1.35) 的更新也可以仅仅追加该迭代所用的向量 y_p 的值到记忆库就行, 因而一次迭代的记忆量的增加也仅是一点点. 但是, 如果照原样执行该方法的话, 随着迭代进行下去

1) 修正单纯形法里也有显式处理基矩阵 B 之逆矩阵的方法. 在这类方法里, 积形式这个词在很多时候是针对用类似式 (1.35) 的形式来直接表现 B^{-1} 而用上的.

必要的记忆量和为解一连串方程式 (1.38), (1.39) 的计算量要变得很大，而且计算误差也累积下去．因此实际上每过几次迭代后有必要把该时刻的基矩阵看成初始基矩阵，重新计算 LU 分解 (1.29)．称这是基矩阵的**再分解** (refactorization).

上述方法看起来复杂，可是因为特别是当用计算机解大规模问题时，在必要的记忆容量和计算误差等方面有利．所以，它已成为现在单纯形法程序的标准想法．

1.6 初始基可行解的计算

1.3 节的单纯形法需要一个基可行解作为初始解． 1.3 节的例 (1.23) 里存在有明显的基可行解，但这仅仅是特别情形．一般问题中必须设法求出初始解．为此，本节介绍经常用到的两阶段法和大 M 法．在以下的讨论中，假设问题已变换成为标准形 (1.1)，而且约束条件右边的常数 b 的所有分量均非负．如果有像 $b_i < 0$ 一样的约束条件，则在该式两边乘以 -1 便可，故这个假定不失一般性．

所谓**两阶段法** (two-phase method)，是首先全力求出问题的一个基可行解，在得到这样的解后，下一步再以它为初始基解计算最优解．两阶段法的第一阶段里，作为针对问题 (1.1) 的辅助问题，考虑如下的线性规划问题．

$$
\begin{aligned}
&\text{目标函数:} d^T y \longrightarrow \text{最小} \\
&\text{约束条件:} Ax + y = b,\ x \geq 0,\ y \geq 0
\end{aligned}
\tag{1.40}
$$

这里，y 为新引进的 m 维变量向量，d 是所有的分量均为 1 的常数向量．变量 y 类似于 1.1 节所述的松弛变量．相比之下，后者的引入为的是把不等式变成等式，它本身具有某种物理意

义, 而前者是为了把本来是等式的约束条件人为变成其它形式的等式条件而用到的 **人工变量** (artificial variable).

考虑对问题 (1.40) 利用单纯形法. 该问题的变量为 (x,y). 划分 y 为基变量, x 为非基变量. 根据假定 $b \geq 0$, 因此 $(x,y)=(0,b)$ 是基可行解. 另外, 这时基矩阵是单位矩阵, 因而对于问题 (1.40), 把该基可行解当作初始解, 就可直接用上单纯形法.

显然, 问题 (1.40) 的目标函数 $d^Ty - \sum_{i=1}^{m} y_i$ 在可行域里取非负值. 于是, 如果最优解是 $(\bar{x},\bar{y})$, 则成立 $d^T\bar{y} > 0$.

如果假定原来的问题 (1.1) 里存在可行解, 则任意可行解 x 和 $y=0$ 的组 $(x,0)$ 满足问题 (1.40) 的约束条件, 其目标函数值 d^Ty 成为 0. 这个事实意味着, 当问题 (1.40) 的最优解 $(\bar{x},\bar{y})$ 里成立有 $d^T\bar{y} > 0$ 时, 原来的问题 (1.1) 没有可行解. 这种情况下, 第一阶段结束时可以结束整个计算

相反, 如果 $(\bar{x},\bar{y})$ 里不成立 $d^T\bar{y} > 0$, 则必然有 $d^T\bar{y} = 0$, 也即成立 $\bar{y}=0$. 在这种情况下, $\bar{x}$ 显然是问题 (1.1) 的一个可行解. 进一步, $(x,y)=(\bar{x},0)$ 是问题 (1.40) 的基可行解. 这时候, 可以认为有人工变量 y_i 全为非基变量以及人工变量中存在有基变量这样两种情况.

在第一种情况下, 基变量全为变量 x 的分量, 故 $x=\bar{x}$ 是问题 (1.1) 的基可行解. 对问题 (1.1) 用单纯形法的时候, 可以把它当作初始基解用.

在第二种情况下, 因为基中留有人工变量, 不能照搬成为问题 (1.1) 的基解. 但是, 这个时候, 通过把进入了基的人工变量 y_i 和满足某条件的非基变量 x_j 交换作转轴运算后, 可以使基变量仅仅由变量 x 的分量构成. 把它当作针对问题 (1.1) 的初始基解即可.

作为例子用两阶段法解如下的问题看一看.

$$
\begin{aligned}
&\text{目标函数:}\quad x_1 - 2x_2 - 3x_3 \longrightarrow \text{最小} \\
&\text{约束条件:}\ 2x_1 + x_2 + x_3 = 6 \\
&\qquad\qquad\quad x_1 + 2x_2 + x_3 = 8 \\
&\qquad\qquad\quad x_1 \geq 0, x_2 \geq 0, x_3 \geq 0
\end{aligned}
\tag{1.41}
$$

第一阶段: 辅助问题 (1.40) 可写成下式. 这里人工变量用 x_4, x_5 表示.

$$
\begin{aligned}
&\text{目标函数:}\ x_4 + x_5 \longrightarrow \text{最小} \\
&\text{约束条件:}\ 2x_1 + x_2 + x_3 + x_4 \qquad = 6 \\
&\qquad\qquad\quad x_1 + 2x_2 + x_3 \qquad + x_5 = 8 \\
&\qquad\qquad\quad x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0
\end{aligned}
\tag{1.42}
$$

初始解

第一步 记初始基矩阵为 $B = [a_4, a_5]$ $x_B = \begin{bmatrix} x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 8 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \end{bmatrix}$

迭代 1

第二步: $w = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $z_1 = [1,1]\begin{bmatrix} 2 \\ 1 \end{bmatrix} = 3$, $z_2 = [1,1]\begin{bmatrix} 1 \\ 2 \end{bmatrix} = 3$, $z_3 = [1,1]\begin{bmatrix} 1 \\ 1 \end{bmatrix} = 2$ 因为 $c_1 - z_1 = -3 < 0$, $c_2 - z_2 = -3 < 0$, $c_3 - z_3 = -2 < 0$, 任意一个非基变量入基都能使目标函数值减少. 这里选取 x_1 为入基变量 $(p = 1)$ 进入第三步

第三步. $y_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $\Delta = \min\{6/2, 8/1\} = 3$ 所以 $r = 1$, 也就是 $x_{B_1}(= x_4)$ 出基. 更新为 $B = [a_1, a_5]$. 新的基变量的值是

$$x_B = \begin{bmatrix} x_1 \\ x_5 \end{bmatrix} = \begin{bmatrix} \Delta \\ 8-\Delta \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$ 回到第二步

迭代 2

第二步. $w = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1/2 \\ 1 \end{bmatrix}$, $z_2 = [-1/2, 1] \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 3/2$

$z_3 = [-1/2, 1] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1/2$, $z_4 = [-1/2, 1] \begin{bmatrix} 1 \\ 0 \end{bmatrix} = -1/2$ $c_2 - z_2 = -3/2 < 0$, $c_3 - z_3 = -1/2 < 0$, $c_4 - z_4 = 3/2 > 0$, 所以变量 x_2 x_3 的任意一个入基都能使目标函数值减少. 这里选取 x_2 为入基变量 $(p = 2)$ 进入第三步.

第三步: $y_2 = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 3/2 \end{bmatrix}$, $\Delta = \min\{3/(1/2), 5/(3/2)\} = 10/3$ 所以 $r = 2$, 也就是 $x_{B_2} (= x_5)$ 要出基. 更新为 $B = [a_1, a_2]$ 新的基变量的值是 $x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 - (1/2)\Delta \\ \Delta \end{bmatrix} = \begin{bmatrix} 4/3 \\ 10/3 \end{bmatrix}$ 回到第二步.

迭代 3

第二步: $w = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $z_3 = [0, 0] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 0$, $z_4 = [0, 0] \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0$, $z_5 = [0, 0] \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0$ $c_3 - z_3 = 0$, $c_4 - z_4 = 1 > 0$, $c_5 - z_5 = 1 > 0$, 所以现在的基解 $x = (4/3, 10/3, 0, 0, 0)$ 是辅助问题 (1.42) 的最优解. 因为人工变量 x_4, x_5 全部成为非基变量, 所以现在的基矩阵 $B = [a_1, a_2]$ 不需变化就可作为第二阶段的初始基使用

第二阶段: 初始解

第一步: $B = [a_1, a_2]$, $x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4/3 \\ 10/3 \end{bmatrix}$ (目标函数值 $z_0 = -8$).

迭代 1

第二步: $w = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $z_3 = [0, -1] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = -1$, 因为 $c_3 - z_3 = -2 < 0$, 故要是让变量 x_3 入基就可以使目标函数值减少.

令 $p = 3$ 进入第二步.

第三步. $y_3 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \end{bmatrix}$,

$\Delta = \min\{(4/3)/(1/3), (10/3)/(1/3)\} = 4$. 所以, $r = 1$, 也即 $x_{B_1}(= x_1)$ 要出基. 更新为 $B = [a_3, a_2]$. 新的基变量的值成为 $x_B = \begin{bmatrix} x_3 \\ x_2 \end{bmatrix} = \begin{bmatrix} \Delta \\ 10/3 - (1/3)\Delta \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$ (目标函数值 $z_0 = -16$). 回到第二步

迭代 2

第二步: $w = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ -2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$, $z_1 = [-4, 1] \begin{bmatrix} 2 \\ 1 \end{bmatrix} = -7$, 因为 $c_1 - z_1 = 6 > 0$, 所以现在的基解 $x = (0, 2, 4)$ 为问题 (1.41) 的最优解

把两阶段法中先求出初始基解后再进入最优解的计算合并成一步进行作业的是下述称为**大 M 法** (big-M method) 的方法. 使用大 M 法时不直接处理问题 (1.1), 而考虑如下的问题.

$$\begin{aligned} &\text{目标函数}: c^T x + M d^T y \quad \longrightarrow \quad \text{最小} \\ &\text{约束条件}: Ax + y = b,\ x \geq 0,\ y \geq 0 \end{aligned} \tag{1.43}$$

这里 M 为正常数, 人工变量 y 及常数向量 d 与两阶段法的一样. 该问题也同样有以人工变量 y 为基变量 (基矩阵为单位矩阵) 的明显基可行解 $(x, y) = (0, b)$, 所以可以把它当作初始解来利用单纯形法.

现在, 设定常数 M 为相当大的值. 这时问题 (1.43) 的目标函数里, 项 $Md^T y (= M \sum_{i=1}^{m} y_i)$ 占支配地位, 所以单纯形法首先让人工变量 y_i 的值减少. 其结果, 如果所有的人工变量 y_i 都成为 0 的话, 目标函数 $c^T x + M d^T y$ 和原来问题 (1.1) 的目标函数 $c^T x$ 本质上一致, 可以认为在此之后单纯形法的迭代所进行

的是求出问题 (1.1) 最优解的计算. 实际上, 问题 (1.1) 有最优解时, 在常数 M 足够大时, 得到的问题 (1.43) 的最优解 $(\bar{x}, \bar{y})$ 中, $\bar{y} = 0$, 而且 $\bar{x}$ 成为问题 (1.1) 的最优解.

但是, 在取定常数 M 解问题 (1.43) 时, 得到的最优解 $(\bar{x}, \bar{y})$ 中可能出现人工变量 $\bar{y}$ 的分量值留有非 0 项. 这时可以认为常数 M 不够大, 所以有必要加大 M 的值, 重解问题 (1.43) 进一步, 在多次重复该操作后人工变量 $\bar{y}$ 仍不为 0 的时候, 判定原来问题 (1.1) 无可行解, 结束计算.

1.7 对偶定理

针对任意一个线性规划问题, 可以定义另一个称为**对偶问题** (dual problem) 的问题. 对偶问题的形式依赖于原问题的形式, 比如, 问题

$$\begin{aligned} &\text{目标函数}: c^T x \longrightarrow \text{最小} \\ &\text{约束条件}: Ax \geq b, \ x \geq 0 \end{aligned} \tag{1.44}$$

的对偶问题如下.

$$\begin{aligned} &\text{目标函数}: b^T w \longrightarrow \text{最大} \\ &\text{约束条件}: A^T w \leq c, \ w \geq 0 \end{aligned} \tag{1.45}$$

比较一下这两个问题, 一方是最小化问题, 另一方是最大化问题. 除此之外还可以从几个地方找出某种对称性. 首先, 一方问题的目标函数系数刚好成为另一方问题的约束条件右边的常数. 这意味着一方问题的变量维数和另一方的约束条件数目相等. 另外, 两个问题的变量都带有非负条件, 但是其它不等式约束条件的不等号的方向相反.

讨论对偶问题的时候，一般把原来的问题称为原问题(primal problem)．然而、这种叫法是相对的．为明确这句话的意思，把上面的问题 (1.45) 看成原问题推导一下它的对偶问题．问题 (1.45) 和

$$
\begin{aligned}
&\text{目标函数:}\quad (-b)^T w \longrightarrow \text{最小}\\
&\text{约束条件:}\quad (-A)^T w \geq \ c,\ w \geq 0
\end{aligned}
$$

等价．根据上面问题 (1.44) 和 (1.45) 的关系，该问题的对偶问题可写成下式．

$$
\begin{aligned}
&\text{目标函数:}\quad (-c)^T x \longrightarrow \text{最大}\\
&\text{约束条件:}\quad (-A)x \leq -b,\ x \geq 0
\end{aligned}
$$

但是因为该问题显然和原问题 (1.44) 等价，由此看出对偶问题的对偶问题和原问题一致．

上面对问题 (1.44) 定义了对偶问题 (1.45) 利用这个关系对任何形式的问题都可以确定其对偶问题的形式．这里，特别以标准形问题 (1.1) 为原问题考虑一下．把问题 (1.1) 改写为问题 (1.44) 的形式，成为

$$
\begin{aligned}
&\text{目标函数:}\quad c^T x \longrightarrow \text{最小}\\
&\text{约束条件:}\quad Ax \geq b,\quad Ax \geq -b,\ x \geq 0
\end{aligned}
$$

因此，根据问题 (1.44) 和 (1.45) 的关系，该问题的对偶问题成为

$$
\begin{aligned}
&\text{目标函数:}\quad b^T w' - b^T w'' \longrightarrow \text{最大}\\
&\text{约束条件:}\quad A^T w' - A^T w'' \leq c,\ w' \geq 0, w'' \geq 0
\end{aligned}
$$

这里，若令 $w = w' - w''$，则最后的问题改写为

$$
\begin{aligned}
&\text{目标函数:}\ b^T w \longrightarrow \text{最大}\\
&\text{约束条件:}\ A^T w \leq c
\end{aligned}
\tag{1.46}
$$

这就是标准形问题 (1.1) 的对偶问题. 注意在标准形问题那样的原问题中，其约束条件为等式时对偶问题的变量不带有非负条件.

这样，给定一个问题可以决定其对偶问题，它们之间有着非常密切的关系. 特别是，记 x 和 w 分别为问题 (1.44) 和 (1.45) 的任意可行解，根据 $Ax \geq b, x \geq 0, A^T w \leq c, w \geq 0$, 成立有

$$c^T x \geq w^T A x \geq w^T b$$

这一关系对任何形式的原 (最小化) 问题和对偶 (最大化) 问题对都成立. 由此可得到如下称作为**弱对偶定理** (weak duality theorem) 的定理[1].

定理 1.4 (弱对偶定理): 任意问题 (P) 及其对偶问题 (D) 之间成立以下关系.

(i) $\left\{\begin{array}{l} x: \text{问题 (P) 的可行解} \\ w: \text{问题 (D) 的可行解} \end{array}\right\} \Longrightarrow c^T x \geq b^T w$

(ii) x: 问题 (P) 的可行解 $\Longrightarrow c^T x \geq$ 问题 (D) 的最大值

(iii) w: 问题 (D) 的可行解 $\Longrightarrow b^T w \leq$ 问题 (P) 的最小值

(iv) $\left\{\begin{array}{l} x: \text{问题 (P) 的可行解} \\ w: \text{问题 (D) 的可行解} \\ c^T x = b^T w \end{array}\right\}$

$\Longrightarrow \left\{\begin{array}{l} x: \text{问题 (P) 的最优解} \\ w: \text{问题 (D) 的最优解} \end{array}\right\}$

(v) 问题 (P) 无界 $\Longrightarrow$ 问题 (D) 无可行解

(vi) 问题 (D) 无界 $\Longrightarrow$ 问题 (P) 无可行解

---

1) 定理 1.4 的 (ii)–(vi) 全部可从 (i) 容易导出，所以有不少数学规划教科书仅仅把 (i) 称为弱对偶定理.

下一步证明原问题有最优解时，对偶问题也一定有最优解．为讨论简单起见，假定原问题是已变换为标准形问题 (1.1) 的问题．设问题 (1.1) 的最优基解为 $x = (x_B, x_N) = (B^{-1}b, 0)$，与它们对应的单纯形乘子是 $w = (B^T)^{-1}c_B$，则根据最优条件 (1.13) 和式 (1.12)，成立有

$$c_N - N^T w \geq 0$$

另外，根据 w 的定义，因为 $B^T w = c_B$，由上式及 $A = [B, N]$，w 应该满足

$$A^T w \leq c$$

也就是，这个 w 是问题 (1.1) 的对偶问题 (1.46) 的可行解．进一步，根据 $w = (B^T)^{-1}c_B$ 和 $x = (x_B, x_N) = (B^{-1}b, 0)$，成立有

$$w^T b = c_B^T B^{-1} b = c_B^T x_B = c^T x$$

因此，由定理 1.4 的 (iv)，w 是对偶问题 (1.46) 的最优解．

像在本节开始所述的那样，原问题和对偶问题的关系是相对的，因此，可以把上面的结果综合成如下的**对偶定理** (duality theorem) 的形式．

定理 1.5 (对偶定理)： 原问题 (P) 和对偶问题 (D) 之中若有一方有最优解则另一方也有最优解．而且这个时候，问题 (P) 的最小值和问题 (D) 的最大值一致．

这里重提前节的例 (1.41)，考虑它的对偶问题．该例是标准形，因此对偶问题可写成下式 (参照式 (1.46))．

$$\begin{aligned}
&\text{目标函数：} 6w_1 + 8w_2 \longrightarrow \text{最大} \\
&\text{约束条件：} 2w_1 + w_2 \leq -1 \\
&\qquad\qquad\quad w_1 + 2w_2 \leq -2 \\
&\qquad\qquad\quad w_1 + w_2 \leq -3
\end{aligned} \tag{1.47}$$

前面已算出，问题 (1.41) 的目标函数最小值为 -16，最优基矩阵是 $B=\begin{bmatrix}1 & 1\\1 & 2\end{bmatrix}$. 这时单纯形乘子成为 $w=\begin{bmatrix}1 & 1\\1 & 2\end{bmatrix}^{-1}\begin{bmatrix}-3\\2\end{bmatrix}=\begin{bmatrix}-4\\1\end{bmatrix}$. 把这个 w 直接代入对偶问题 (1.47)，显然满足所有的约束条件，而且它对应的目标函数值是 -16. 于是它是对偶问题的最优解.

1.8 灵敏度分析

现实问题里，有很多时候是基于统计方法及预测等等得到的数据来决定系数. 在这种情况下，不仅要考虑系数的某个特定值，而且有必要对实际上可能出现的各种各样的值来进行综合考虑. 所谓**灵敏度分析** (sensitivity analysis)，就是利用解一个问题得到的结果，研究当系数有微小变化时最优解的反应. 这是详细分析问题所不可欠少的工具.

为简单起见，考察问题 (1.1) 里约束条件右边常数向量 $b=(b_1,b_2,\dots,b_m)^T$ 作微小变化后，成为 $b+\Delta b=(b_1+\Delta b_1,b_2+\Delta b_2,\dots,b_m+\Delta b_m)^T$ 后的问题 [1]

$$\begin{aligned}&\text{目标函数}: c^Tx \longrightarrow \text{最小}\\&\text{约束条件}: Ax=b+\Delta b,\ x\geq 0\end{aligned}\tag{1.48}$$

现在，假设对问题 (1.1) 利用单纯形法得到最优基解 $x^*=(x_B^*,x_N^*)=(B^{-1}b,0)$. 这时对于最优基 B，根据可行性条件

$$B^{-1}b\geq 0\tag{1.49}$$

1) 目标函数的系数向量 c 变化的情况，可以归结为考虑对偶问题的约束条件右边的常数向量变化的情况. 但是，约束条件左边的系数矩阵 A 变化的情况比起 b 和 c 的情况叙述起来要复杂得多.

根据最优性的条件

$$C_N^T - C_B^T B^{-1} N \geq 0 \tag{1.50}$$

(参照式 (1.12), (1.13)). 这里考虑让问题 (1.1) 里的 b 值变化，式 (1.49) 里的 b 换成 $b + \Delta b$ 后如果成立

$$B^{-1}(b + \Delta b) \geq 0 \tag{1.51}$$

的话，因式 (1.50) 和 b 没关系，故知道针对问题 (1.1) 的最优基 B 也是针对问题 (1.48) 的最优基. 特别是，问题 (1.1) 的最优解满足非退化条件

$$B^{-1}b > 0$$

时，式 (1.51) 对足够小的任意 Δb 成立. 也就是，系数 b 的变化量很小的时候，最优基无变化，问题 (1.48) 的最优解成为 $x = (x_B, x_N) = (B^{-1}(b + \Delta b), 0)$. 进一步，把问题 (1.1), (1.48) 的目标函数最小值分别记为 $z(b)$, $z(b + \Delta b)$. 因有

$$z(b) = c_B^T B^{-1} b$$
$$z(b + \Delta b) = c_B^T B^{-1}(b + \Delta b)$$

故成立

$$z(b + \Delta b) - z(b) = c_B^T B^{-1} \Delta b$$

但是 $(B^T)^{-1} c_B$ 是对应基 B 的单纯形乘子 w, 根据前节的讨论，它是问题 (1.1) 的对偶问题的最优解. 因此，若记 $w^* = (B^T)^{-1} c_B$, 则上式可改写成为

$$z(b + \Delta b) - z(b) = (w^*)^T \Delta b = \sum_{i=1}^{m} w_i^* \Delta b_i$$

进一步如果考虑 $\Delta b_i \to 0$ 时的极限，就可以得到

$$\frac{\partial z(b)}{\partial b_i} = w_i^* \quad (i = 1, 2, \dots, m) \tag{1.52}$$

这是对偶问题的最优解 w^* 的第 i 分量 w_i^*，证明它等于问题 (1.1) 里第 i 个约束条件右边在 b_i 基础上仅仅增加 1 单位时目标函数最小值所增加的比例.

这里，考虑上面所述事实的经济意义.

假设问题 (1.1) 是利用所给定的几种资源，使生产成本达最小的生产问题模型. 这里，b_i 是第 i 种资源的可能利用量，$z(b)$ 表示的是这时对应于最优生产的成本. 根据式 (1.52)，因为 w_i^* 等于第 i 种资源的可能利用量发生变化时对最优生产成本的影响量. 它反应了第 i 种资源在该系统里具有的潜在价值. 在这意义下，称 w_i^* 为第 i 资源的**潜在价格** (shadow price 或**影子价格**). 一般说来，各资源的系统内的价值 (潜在价格) 依赖于全资源的目前可能利用量 b. 它与在系统外部的价值（市场价格）不是一个东西. 比如，在有可投于生产活动的剩余资金的时候，比较各资源的潜在价格和市场价格，可以认为筹集其差较大的资源是比较有效果的.

1.9 文献及其它话题

单纯形法作为研究对象来说几乎到了成熟的地步，为数众多的优秀著作在国内外出版. 这里除单纯形法的生父 Dantzig 所著的 [D63] 外，还列举 Chvátal, [C83], 伊理 [I86], 古林 [K80], 今野 [K87]. 特提一下，Chvátal [C83] 清楚明了地记述了从线性规划的理论到应用的广泛内容，是本好著作. 建议想更详细了解本章所述内容的读者读一读这本书. 另外，作为综述了线性规

划从单纯形法到最近的内点法的演变过程的论文，有 Goldfarb, Todd [GT89].

1.3 节里讲过，单纯形法里除修正单纯形法外还有利用单纯形表的执行法. 后面的方法在上述教科书里也出现了. 特别是茨木，福岛 [IF91] 里还有附带着 FORTRAN 程序的详细讲解.

1.4 节所讲到的最小下标规则 (定理 1.3) 是根据 Bland [B77] 所写的. 本书里省略了对最小下标规则正确性的数学证明. 有兴趣的读者请参照原论文或者上述教科书 [C83,I86,K87]. 另外，像正文中提及的那样，从最坏计算量角度看单纯形法不是多项式时间算法，但是经验告诉我们它是足够实用的. 关于单纯形法的实际及理论上的计算效率的综述论文有 Shamir [S87].

随着单纯形法及内点法的进步，现在实用上可求解规模相当大的线性规划问题. 比如， Beasley [B90] 报告了对有数万到数十万个变量的问题用单纯形法的数值实验结果. 另外，根据把单纯形法和内点法进行组合的方法，对航空公司乘务员从排序问题派生出来的带有约 1, 275 万个变量的线性规划问题 (这里系数矩阵的分量为 0 或 1) 有用超级计算机在约 4 分钟内解出的报告 [BGLMS92].

第 2 章 网络最优化

最短路问题，最大流问题，最小成本流问题等有关网络的最优化问题在应用场合下也经常出现．这些问题之中大部分可模型化成为线性规划问题的特殊情况，所以根据针对线性规划问题的一般结果，可在多项式时间内解出．但是，如果利用问题的特殊性，有可能更快解出．本章在综观整个网络最优化之后，稍微详细介绍针对最小成本流问题的强多项式时间算法．

2.1 网络最优化问题

本章里，$G = (V, E)$ 为有向图[1]．有时候把其中某个节点作为**始点** s，另一个作为**终点** t 而特殊对待．各边 $e \in E$ 上赋有**成本** (cost) $c(e)$ 以及**容量** (capacity) $u(e)$．$c(e)$ 以及 $u(e)$ 都取实数值，把这些元素组成的 $|E|$ 维向量分别记为 $c = (c(e); e \in E)$，$u = (u(e); e \in E)$．通常假定 $u(e) \geq 0$ $(e \in E)$，但 $c(e)$ 的符号是自由的．统括这些的

$$N = (G, s, t, c, u)$$

称为**网络** (network) (在不同的场合下可能只处理 s, t, c, u 的部分)．另外，$e = (v, w)$ 的时候，也可以把 $c(e)$ 及 $u(e)$ 记为 $c(v, w)$ 及 $u(v, w)$．

以下介绍几个现实应用中经常出现的网络最优化问题．

---

1) 有关图和网络的术语请参照附录 A.3

2.1.1 最短路问题 (shortest path problem)

在最短路问题中，成本 $c(e)$ 解释为边 e 的长度．最短路问题的典型形式是，在网络 $N=(G,s,c)$ 里求出始点 s 到其它各点 $v\in V$ 的最短路及其长度．这里，假设从 s 可以到所有其它的点 $v\in V$．另外，定义 $w\in V$ 到 $v\in V$ 的路 (path)

$$\pi = v_{i_0}(=w), v_{i_1}, v_{i_2}, \ldots, v_{i_k}(=v) \tag{2.1}$$

的长度为

$$c(\pi) = \sum_{j=0}^{k-1} c(v_{i_j}, v_{i_{j+1}}) \tag{2.2}$$

当 N 中没有长度为负的回路的时候，存在有从 s 到所有点 v 的最短路 (如果存在有负的回路，则通过多次迂回该回路可使路长发散至 ∞)，它们可以用以 s 为根的**最短路树**来表示．图 2.1 的例中，粗边表示最短路树．求 s 到 v 的最短路时，只要沿着 s 到 v 的粗边走下去就行．

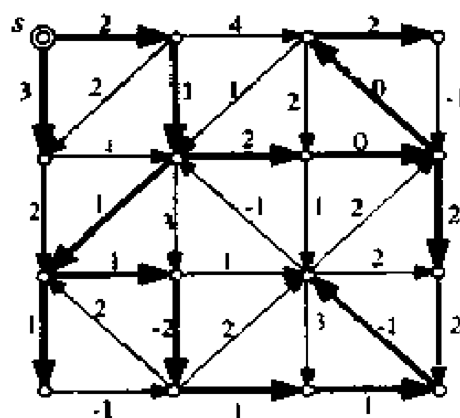


图 2.1 网络和最短路树

当 G 是无向图的时候也可以同样定义最短路问题．这种情况下，两条边 (v,w) 和 (w,v) 不加以区别，所以认为边长满足 $c(v,w)=c(w,v)$，上面关于最短路的性质同样成立．

2.1.2 最大流问题 (maximum flow problem)

网络 $N=(G,s,t,u)$ 里，所谓从始点 s 到终点 t 的**流** (flow) 是指满足下面条件的实数向量 $x=(x(e);e\in E)$

$$\sum_{e\in \mathrm{OUT}(v)} x(e) \quad \sum_{e\in \mathrm{IN}(v)} x(e)=0 \quad (v\in V-\{s,t\}) \tag{2.3}$$

这里 $\mathrm{OUT}(v)$ ($\mathrm{IN}(v)$) 表示 G 中流出 (流入) v 的边的集合．条件 (2.3) 表示在 s 和 t 以外的各节点 v 上，从 v 流出的流量之和 $\sum_{e\in \mathrm{OUT}(v)} x(e)$ 与流入 v 的流量之和 $\sum_{e\in \mathrm{IN}(v)} x(e)$ 相等，称为**流量守恒条件** (flow conservation constraint)．结果，从 s 的纯流出量和到 t 的纯流入量

$$x(s) \triangleq \sum_{e\in \mathrm{OUT}(s)} x(e) \quad \sum_{e\in \mathrm{IN}(s)} x(e)$$
$$x(t) \triangleq \sum_{e\in \mathrm{IN}(t)} x(e) - \sum_{e\in \mathrm{OUT}(t)} x(e)$$

相等，从而成立

$$x(s) - x(t) \tag{2.4}$$

这个 $x(s)$ (也即 $x(t)$) 称为 x 的**流值** (flow value)

当流 x 进一步满足**容量条件** (capacity constraint)

$$0\le x(e) < u(e) \quad (e\in E) \tag{2.5}$$

的时候，x 称为**可行流** (feasible flow) 所谓最大流问题，就是

求出使流值达最大的可行流问题，可描述如下.

$$
\begin{aligned}
&\text{目标函数:}\ x(s) \longrightarrow \text{最大}\\
&\text{约束条件:}\ \sum_{e\in \mathrm{OUT}(v)} x(e) - \sum_{e\in \mathrm{IN}(v)} x(e) = 0 \quad (v \in V - \{s,t\})\\
&\qquad\qquad x(s) = \sum_{e\in \mathrm{OUT}(s)} x(e) - \sum_{e\in \mathrm{IN}(s)} x(e) \qquad\qquad (2.6)\\
&\qquad\qquad 0 \le x(e) \le u(e) \quad (e \in E)
\end{aligned}
$$

2.1.3 最小成本流问题 (minimum cost flow problem)

考虑网络 $N = (G, s, t, c, u)$ 上的流 x. 这里，固定流值为

$$x(s) = f^*$$

当然，该 f^* 的值必须在 N 的最大流的值以下. 在这个条件下，求出使成本达最小的流就是最小成本流问题.

$$
\begin{aligned}
&\text{目标函数:}\ \sum_{e\in E} c(e)x(e) \longrightarrow \text{最小}\\
&\text{约束条件}\ \sum_{e\in \mathrm{OUT}(v)} x(e) - \sum_{e\in \mathrm{IN}(v)} x(e) = 0 \quad (v \in V - \{s,t\})\\
&\qquad\qquad \sum_{e\in \mathrm{OUT}(s)} x(e) - \sum_{e\in \mathrm{IN}(s)} x(e) = f^* \qquad\qquad (2.7)\\
&\qquad\qquad 0 \le x(e) < u(e) \quad (e \in E)
\end{aligned}
$$

该模型里，始点以及终点都仅仅是一个. 但是，这并不是本质性的约束. 因为如果要解的最小成本流问题具有 k 个始点 s_j, $j = 1, 2, \cdots, k$ 以及各自的流出量 $x(s_j) = f^*(s_j)$, 还有 k' 个终点 t_j, $j = 1, 2, \cdots, k'$ 以及各自的流入量 $x(t_j) = f^*(t_j)$, 则可以导入 (虚设) 哑元始点 s 和哑元终点 t, 加上 $(k + k')$ 条边

(s, s_j), 成本 $c(s, s_j) = 0$, 容量 $u(s, s_j) = f^*(s_j)$ $(j = 1, 2, \cdots, k)$

(t_j, t), 成本 $c(t_j, t) = 0$, 容量 $u(t_j, t) = f^*(t_j)$ $(j = 1, 2, \cdots, k')$

后，求出具有流值

$$f^* = \sum_{j=1}^{k} f^*(s_j) \quad (= \sum_{j=1}^{k'} f^*(t_j)) \tag{2.8}$$

的从 s 到 t 的最小成本流就行（参照图 2.2 的虚线部分）

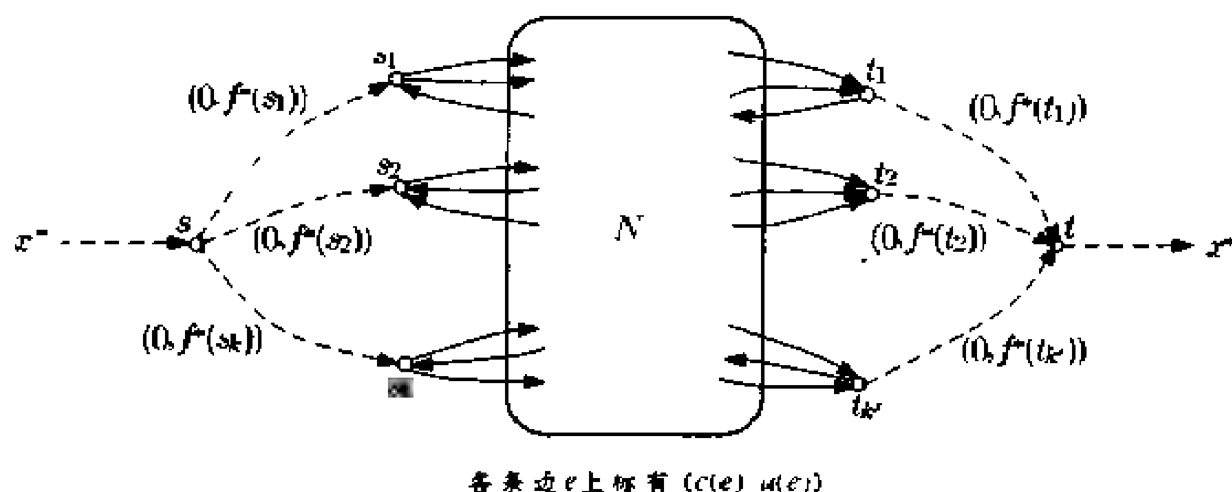


图 2.2　具有多个始点和终点的最小成本流问题的转化（加上虚线部分）

在这里，我们证明最小成本流问题作为其特殊情况包含有最短路问题以及最大流问题．首先，通过导入哑元终点 t，加上从 $v \in V$ 出发的边

$$(v, t)\ ;\ c(v, t) = 0, u(v, t) = 1$$

后，最短路问题成为解从 s 到 t 的具有流值 $f^* = n$ 的最小成本流问题．这里，$n = |V|$，另外原网络内的边的容量仓假定为 ∞．把所得到的流[1] x 分解成从 s 出发经各点 v 到达 t 的 n 个流（各自都具有值 1），追寻每个流得到的路（除去边 (v, t) 后）就给出从 s 出发的最短路．

1) 最小成本流问题里，当容量 $u(e)$ $(e \in E)$ 为整数时，可以不失一般性而假定 $x(e)$ $(e \in E)$ 也是整数

其次，关于最大流问题，在 N 中导入哑元始点 s'，加上两条边

$$(s', s);\quad c(s', s)\quad 0,\quad u(s', s) = \infty$$
$$(s', t);\quad c(s', t) = 1,\quad u(s', t)\quad \infty$$

后，求解从 s' 到 t 的最小成本流问题即可．这里，设定在原网络部分里 $c(e) = 0$ $(e \in E)$ 并取最大流值的适当的上界 (比如说 $\sum_{e\in E} u(e)$) 为从 s' 出发的流值 f^*.

得到的最小成本流 x 里，容易证明 $x(s', s)$ 给出原问题的最大流值．

2.1.4 最小成本循环流问题 (minimum cost circulation problem)

不像最小成本流问题那样假定特定的节点为始点或者终点，现在考虑在各边 e 上导入流的下界值 $l(e)$ 而得到的网络 $N\quad (G, c, l, u)$. 在这个网络上求出使成本和最小的流的问题称为最小成本循环流问题．

$$\text{目标函数：} \sum_{e\in E} c(e)x(e) \longrightarrow \text{最小}$$

$$\text{约束条件：} \sum_{e\in \mathrm{OUT}(v)} x(e) \quad \sum_{e\in \mathrm{IN}(v)} x(e) = 0 \quad (v \in V) \tag{2.9}$$
$$l(e) \le x(e) < u(e) \quad (e \in E)$$

$l(e)$ 和 $u(e)$ 的值可以为满足

$$l(e) \le u(e) \quad (e \in E) \tag{2.10}$$

的任意实数，也有为负的时候．

和最小成本流问题 (2.7) 相比，问题 (2.9) 的约束条件更简单，因而具有可以清晰描述其算法等优点．但是如下所示，其

模型化能力和最小成本流问题等价.

首先，关于从 s 到 t 的值为 f^* 的流之最小成本流问题 (2.7) 可以这样转化：设计一条 t 到 s 的边

$$(t,s); c(t,s)=0, l(t,s)=f^*, u(t,s)=f^*$$

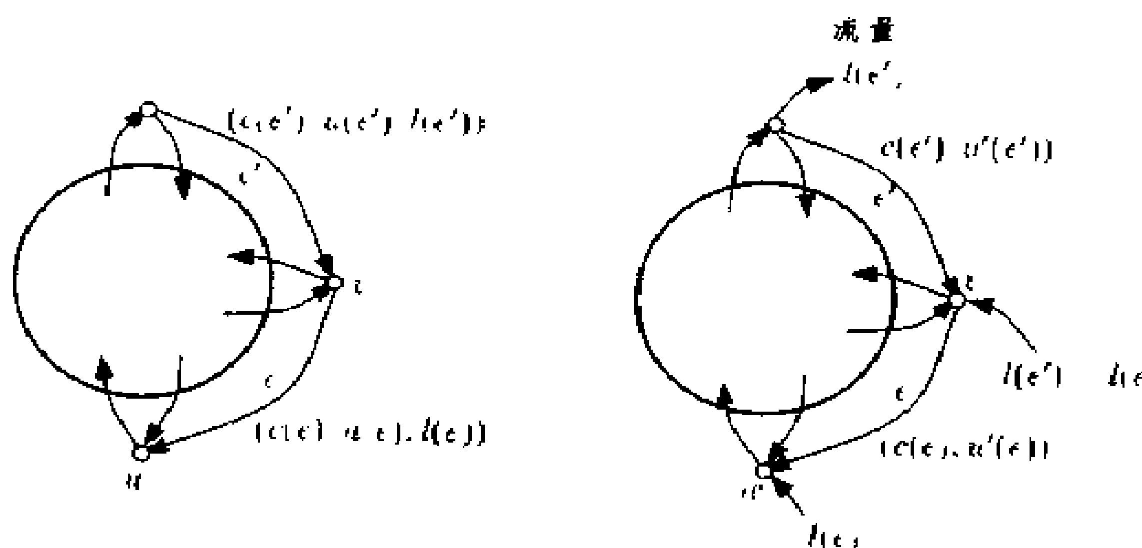
(网络其它部分保持不变), 所得到的网络的最小成本循环流问题即为 (2.7). 这是因为在边 (t,s) 上设定了 $l(t,s)=u(t,s)$, 所以由在点 s 的流量守恒条件, $x(s)=f^*$ 得以实现.

另一方面，在任意的最小成本循环流问题里，针对满足 $l(v,w)\neq 0$ 的各边 (v,w) 进行如下操作，

1. 除去 (v,w) 的下界值 $l(v,w)$ 并令上界值 $u'(v,w)=u(v,w)-l(v,w)$,

2. 把 w 看成具有流出量 $l(v,w)$ 的始点，

3. 把 v 看成具有流入量 $l(v,w)$ 的终点 (即流出量为 $-l(v,w)$ 的始点), 如图 2.3,



(a) 最小成本循环流问题　　(b) 最小成本流问题

图 2.3　最小成本循环流问题转化为最小成本流问题

(仅表示出与边 e 和 e' 相关联的部分)

其结果，从各节点 v 的流出量成为

$$l(v) \quad \sum_{e\in \mathrm{IN}(v)} l(e) - \sum_{e\in \mathrm{OUT}(v)} l(e)$$

这里，满足 $l(v) > 0$ 的 v 改为有值 $f^*(v) - l(v)$ 的始点，满足 $l(v) < 0$ 的 v 改为有值 $f^*(v) \quad -l(v)$ 的终点. 得到的网络虽然有多个始点以及终点，但是像 2.1.3 所述的那样，也可以把这个问题当作最小成本流问题. 最后，基于得到的最优流 x, 令

$$x'(e) = x(e) + l(e) \quad (e \in E)$$

则 x' 给出最小成本循环流问题的最优解.

像已经讲过的那样，最小成本流问题 (以及与之等价的最小成本循环流问题) 包含有其特殊情形：最大流问题. 另一方面，最大流问题可以用来得到最小成本流 (以及最小成本循环流问题) 的可行流. 也即，在最小成本流问题 (2.7) 里，设定一个哑元节点 s' 和边

$$(s', s)\ ;\ u(s', s) = f^*$$

如果求出了 s' 到 t 的最大流 x, 则在 x 满足 $x(s', s) = f^*$ 的时候，它即成为原来问题的可行流. 与此相反，如果 $x(s', s) < f^*$, 则该最小成本流问题没有可行流. 这个性质经常用来在最小成本流问题和最小成本循环流问题中有效地求出可行流.

本章在下一节综述这些网络最优化问题算法的进展之后，在 2.3 节 ~ 2.5 节中详细叙述针对最一般的问题，即最小成本循环流问题的算法.

2.2 网络最优化算法的进展

前节所述的网络最优化问题全部具有线性规划的形式，可以用第 1 章的单纯形法求解．但是，它们都具有基于网络的特殊结构，所以相应地把单纯形法进一步特殊化后有可能提高计算效率．事实上，人们知道，基于图 G 的生成树，利用简便的数据结构可以保持这些问题的基解．为此，具有 n 个节点 m 条边的网络需要的空间量是 $O(m+n)$ (如果用通常的列表单纯形法则需要 $O(mn)$ 的空间)．其结果，得到了**网络单纯形法** (network simplex method)．可以认为这种方法是在实用上是最快的算法，尤其是对于最小成本流问题．(但是，像本章后半部分要讲的那样，理论上人们的兴趣集中在另外一个算法上．)

本书第 5 章要讲到，任意的线性规划问题都可在多项式时间内解出[1]．于是，网络最优化问题也全是如此．这里，请回忆所谓多项式时间是以输入数据长度为基准的．也就是说，给定节点数 n，边数 m，成本系数的最大值 $C=\max_e|c(e)|$，以及容量的最大值 $U=\max_e|u(e)|$ 的时候，其计算时间量是 n, m 以及 $\log C$ 和 $\log U$ 的多项式阶的意思 (另外，为简单起见，假定了 $c(e), u(e)$ 全是整数)．因此，在时间量的评价中，除去 $\log C$ 和 $\log U$，如果可以用仅含 n 和 m 的多项式控制，则称这样的算法是**强多项式时间** (strongly polynomial time) **算法**．因不依赖于 C 和 U 的大小，可以认为这个评价是更满意的．

关于线性规划问题，总体上是否存在强多项式算法的问题至今尚未解决．但是对一些网络最优化问题已知道有强多项式算法．关于最短路问题和最大流问题，本节以下作简单说明．关于最小成本流问题，本节介绍多项式算法，在下一节以后稍

1) 关于算法计算量的定义请参照附录 A 4.

微详细讲述强多项式算法. 最小成本流问题的强多项式时间算法是比较新的话题.

2.2.1 最短路问题的算法

关于这个问题, 基于**动态规划法** (dynamic programming) 的思想, 人们很早就知道有几个强多项式时间算法. 以下说明具有代表性的算法之一、**戴克斯特拉法** (Dijkstra's algorithm).

计算中求出的 $d(v)$, 恒为从 s 到 v 的最短路长 $c^*(v)$ 的上界, 成立有 $d(v) \geq c^*(v)$ 但在节点集合 $P \subset V$ 里, 有 $d(v) = c^*(v)$ $(v \in P)$ 的性质. 于是, 当 $P = V$ 的时候, 求出了到所有节点的最短路路长、结束计算.

算法 DIJKSTRA (戴克斯特拉法)

输入: 有向图 $G = (V, E)$, 边长 $c : E \to R_+$ (这里 R_+ 是非负实数的集合), 始点 $s \in V$

输出: 到所有的点 $v \in V$ 的最短路长 $c^*(v)$.

第一步 (初始化): 令 $d(s) := 0$ 以及 $d(v) := \infty$ $(v \in V - \{s\})$. $P := \emptyset$.

第二步 (迭代): 选取一个满足 $d(v^*) = \min\{d(v) | v \in V - P\}$ 的 $v^* \in V - P$

第三步 (更新): $c^*(v^*) := d(v^*)$, $P := P \cup \{v^*\}$, 进一步, 对满足 $w \in V - P$ 的各边 $e = (v^*, w) \in E$ 作如下更新

$$d(w) := \min\{d(w), d(v^*) + c(e)\}$$

第四步 (结束判定): 如果 $P = V$ 则结束计算. 如果 $P \neq V$ 则回到第二步.

在戴克斯特拉法里, 边长 $c(e)$ 的非负性是本质的. 如果存在有负边, 则运行情况可能不佳. 另外, 不失一般性, 假定 s 到其它节点 v 至少都存在有一条路 (可达)

戴克斯特拉法的时间量, 因实现第二步和第三步的数据结构不同而不同. 首先, 第二步和第三步在时间 $O(n)$ 内是容易执行的. 因为迭代要进行 n 次, 所以这种情况下的总时间量成为 $O(n^2)$. 另外, 如果保持 $d(v)$ $(v \in V - P)$ 的堆形数据结构, 则第二步可在常数时间里执行而第三步对每条边 e 可以在 $O(\log n)$ 时间里执行. 因此, 总时间量是 $O(m \log n)$. 任何一种都是强多项式时间.

因篇幅的限制, 在此省略戴克斯特拉法能正确运行的证明, 也省略和戴克斯特拉法不同的最短路算法的说明. 请参照其它文献. 在 "阶" 的意义上, 现在最快的是 Fredman 和 Tarjan 的算法 [FT 87], 其阶是 $O(m + n \log n)$.

2.2.2 最大流问题的算法

网络 $N = (G, s, t, u)$ 里给定了可行流 x 的时候, 如下定义**余网络** (residual network) N_x 和**流扩充路** (flow augmenting path) 从 N 的各边 $e = (v, w) \in E$ 出发, 按以下规则生成边 (v, w) 和 (w, v) (根据情况仅取一方) 得到的边集合记为 E_x. 与此同时, 确定容量 u'.

规则 1: 如果 $u(e) - x(e) > 0$, 则生成 $(v, w) \in E_x$, 并令其容量 $u'(v, w) = u(e) - x(e)$.

规则 2: 如果 $x(e) > 0$, 则生成 $(w, v) \in E_x$ 并令其容量 $u'(w, v) =$

$x(e)$.

所得到的有向图 $G_x = (V, E_x)$ 和容量 u' 合起来便是余网络 $N_x = (G_x, s, t, u')$. N_x 中从 s 到 t 的路称为流扩充路.

引理 2.1 (流扩充路) 给定网络 $N = (G, s, t, u)$. 其可行流 x 为最大流的充分必要条件是，余网络 N_x 里不存在有流扩充路.

根据下面的事实很容易理解这个引理的必要性. 如果存在有流扩充路，则 x 可以修正为流值比它更大的流 x'. 也就是，假定在 N_x 里存在 s 到 t 的路 π. 则利用

$$\Delta := \min\{u'(e) | e \in \pi\} \tag{2.11}$$

对各边 $e = (v, w) \in E$ 作如下修正

$$x'(e) := \begin{cases} x(e) + \Delta, & e \in \pi \text{ (} e \text{ 是根据规则 1 生成的边) 的情况下} \\ x(e) - \Delta, & e' = (w, v) \in \pi \text{ (} e' \text{ 是根据规则 2 生成的边) 的情况下} \\ x(e), & \text{其它情况下} \end{cases} \tag{2.12}$$

这里，$e = (v, w) \in \pi$ 意味着路 π 从节点 v 进入到下一个节点 w，把 π 看成是形成该路的边的集合. 这个时候，很容易证明，x' 也是 N 的可行流，其值是

$$x'(s) = x(s) + \Delta$$

仅增加了 $\Delta\ (> 0)$. 为证明引理的充分性，需要稍深的讨论，在此省略.

利用这个关于流扩充路的性质，可以得到如下求最大流的算法.

算法 MAX-FLOW (最大流)

输入：网络 $N=(G=(V,E),s,t,u)$

输出：从 s 到 t 的最大流值 $f_{\max}$

第一步 (初始化)：令 $f_{\max}:=0$ 以及 $x(e):=0\ (e\in E)$.

第二步 (余网络)：构造余网络 $N_x=(G_x,s,t,u')$

第三步 (流的扩充)：如果 N_x 中没有流的扩充路则结束计算. 相反，如果存在有流的扩充路，则选取其一 π，根据式 (2.11) 和式 (2.12) 修正 x 得到流 x'，再把它当成 x，令 $f_{\max}:=f_{\max}+\Delta$. 回到第二步.

不失一般性，假定图 G 里从 s 可以通达所有其它节点. 很容易知道第二步里构造余网络能够在 $O(m)$ 时间内完成. 求出流的扩充路从而修正 x 的第三步也能够在 $O(m)$ 时间内完成. 如果考虑容量 $u(e)$ 全为非负整数的情况，则式 (2.11) 的 Δ 是 1 以上的整数，因而，每次迭代流的值增加 1 以上. 把最大流的值记为 $f_{\max}$，则迭代次数在 $f_{\max}$ 以下，该算法的时间量是 $O(mf_{\max})$.

很遗憾，$O(mf_{\max})$ 并非输入数据长 $n+m\log U$ (这里 $U=\max_e u(e)$) 的多项式阶，另外，因包含有 (计算开始时的) 未知参数 $f_{\max}$ 而不太令人满意. 因此，不断有人尝试着把算法 MAX-FLOW 修正为强多项式算法. 其中之一是下面 Edmonds Karp [EK 72] 的结果.

MAX-FLOW 的第二步里选取流的扩充路 π 的时候，选取边数最小者．如果在 N_x 里利用宽度优先的探索法，则这样的 π 可以在 $O(m)$ 时间内求出．式 (2.11) 里，把满足 $u'(e) = \Delta$ 的 $e \in \pi$ 称为瓶颈边．成立有下面的性质．

- 迭代过程中，同一条边 (v, w) 又一次成为瓶颈边的话，则一定在中途的某次迭代里 (w, v) 成为瓶颈边．
- (v, w) 之后 (w, v) 首先成为瓶颈边的时候，从 s 到 t 的最小边数的路所包含的边数至少增加 2.

考虑 N_x 里从 s 到 t 的最小边数的路，它所包含的边的数目一定在 $n-1$ 以下．因此，瓶颈边的序列中 (v, w) 和 (w, v) 出现的次数和最多是 $(n-1)/2+1=(n+1)/2$. 同样的讨论对所有的边 $(v, w) \in E$ 成立．这就推出，在这个选择规则下，MAX-FLOW 的迭代次数在 $(n+1)m/2$ 以下．因此，这种情况下的计算量成为 $O(nm^2)$，是多项式时间．

此后，改善时间量的努力还在进一步继续，主要在以下几个方面下了功夫：

- 改变选取流扩充路的基准
- 在一次迭代中求出多个流的扩充路
- 在每次迭代中将得出可行流的条件加以放松，利用称为预流 (preflow)[1] 的解
- 改良数据结构．

1) 满足容量约束及缓和流量守恒条件 $\sum_{e\in \mathrm{OUT}(v)} x(e) - \sum_{e \in \mathrm{IN}(v)} x(e) \le 0$ $(v \in V)$（也即，在各节点 v 处的流入量在其流出量之上）的流 x 称为预流

表 2.1 概括了这段时间里的发展情况.

表 2.1 最大流问题的算法的进展(这里 n = 节点数,

m = 边数, $f_{\max}$ = 最大流值, $U = \max_e u(e)$)

发现者	时间量
Ford, Fulkerson (1956)[FF62]	$O(mf_{\max})$
Edmonds, Karp (1969)[EK72]	$O(nm^2)$
Dinic (1970)	$O(n^2m)$
Karzanov (1974)	$O(n^3)$
Cherkasky (1977)	$O(n^2\sqrt{m})$
Shiloach (1978)	$O(mn\log^2 n)$
Galil, Naamad (1979)	$O(mn\log^2 n)$
Galil (1980)	$O(n^{5/3}m^{2/3})$
Sleater, Tarjan (1983)	$O(mn\log n)$
Gabow (1985)	$O(mn\log U)$
Goldberg, Tarjan (1986)[GT88]	$O(mn\log(n^2/m))$
Ahuja Orlin (1987)	$O(mn + n^2\log U)$
Ahuja, Orlin, Tarjan (1987)	$O\left(mn\log\left(\frac{n\log U}{m\log\log U}+2\right)\right)$

2.2.3 最小成本流问题的算法

对该问题最初的多项式算法由 Edmonds 和 Karp [EK72] 提出, 它成了以后发展强多项式时间算法的开端. 下面简要介绍该算法.

与上面的最大流问题一样, 定义余网络 $N_x = (G_x, s, t, c', u')$ 这里, c' 是如下确定的成本系数向量:

$$c'(v,w) = c(v,w), \quad \text{当}(v,w) \in E_x \text{由规则 1 生成的时候},$$
$$c'(w,v) = -c(v,w), \quad \text{当}(w,v) \in E_x \text{由规则 2 生成的时候}.$$

引理 2.2 (最小成本流的增量法) 网络 $N = (G, s, t, c, u)$

里，流值为 a 的最小成本流记为 x^a 这个时候，余网络 N_{x^a} 里求出 s 到 t 的最短路 π (各边 e 的长度为 $c(e)$)，如果根据式 (2.11)，(2.12) 修正流 x^a，则得到的流 x^b 是对应流值 $b = a + \Delta$ 的最小成本流.

假定网络 N 没有成本和为负的回路. 这个时候， $x = 0$ 是对应流值 $a = 0$ 的最小成本流. 于是，从 $x = 0$ 开始，反复利用上面的引理增加流值，就可以求出对应所要求的流值的最小成本流.

算法 MINCOST (最小成本流)

输入：网络 $N = (G = (V, E), s, t, c, u)$ 以及流值 f^* (> 0).

输出：最小成本流 x

第一步 (初始化)[1]：$x := 0, a := 0$.

第二步 (余网络)：构造余网络 $N_x = (G_x, s, t, c', u')$

第三步 (增加流量)：如果 N_x 没有 s 到 t 的路，则结束计算. N 中不存在有流值为 f^* 的可行流. 否则，求出 s 到 t 的最短路 π，计算 (2.12) 的 Δ.

(1) 如果 $a + \Delta \geq f^*$，则改令 $\Delta := f^* - a$，根据 (2.12) 求出 x'，令 $x = x'$ 结束计算. x 是要求的最小成本流.

(2) 如果 $a + \Delta < f^*$，则根据 (2.12) 求出 x'，令 $x := x'$，$a := a + \Delta$ 后回到第二步.

---

1) 对向量 x，记号 $x = 0$ 表示对 x 的所有分量有 $x(e) = 0$.

该算法的主要计算部分是第一步中 N_x 的构造，第二步中最短路 π 的计算和 x 的修正. N_x 的构造和 x 的修正显然在 $O(m)$ 的时间内可以完成. 在计算最短路的时候，如果假定 N 中不存在有成本和为负的回路，则能够事先作变换使得 N_x 里所有的 $c'(e)$ 非负 (在此省略其详细介绍[1]). 因此，根据 2.2.1 的讨论，在 $O(m+n\log n)$ 时间内 (用 Fredman 和 Tarjan 的算法) 可以求出 π. 在 u 为整数向量的前提下，因为 $\Delta \geq 1$，故可以推出第二步和第三步的迭代次数在 f^* 以下. 于是，整个时间量是

$$O((m+n\log n)f^*) \tag{2.13}$$

可是，这还不是多项式时间.

为得到多项式时间算法，下面采用的方法是对 u 的**度量法** (scaling method). 也就是当 $U=\max_e u(e)$ 满足 $2^p \leq U < 2^{p+1}$ (也即 $p=\lfloor \log_2 U \rfloor$)[2] 的时候，把 $u(e)$ 及 f^* 舍为

$$\begin{aligned} u^{(k)}(e) &= \lfloor u(e)/2^{p-k} \rfloor \\ f^{*(k)} &= \lfloor f^*/2^{p-k} \rfloor \end{aligned} \tag{2.14}$$

这也就相当于，把系数 $u(e)$ 和 f^* 表示为 $p+1$ 位 2 进制的时候，只从大的方向取 $k+1$ 位来表示. 度量法对

$$k=0,1,\cdots,p$$

依次求解如此得到的网络 $N^{(k)}=(G,s,t,c,u^{(k)})$ 中具有流值 $f^{*(k)}$ 的最小成本流问题.

1) 采取和后面所述引理 2.4 中 (ii) ⇒ (iii) 的证明同样的方法，利用现在得到的到各节点 v 的最短路长，可以在 $O(n)$ 时间内完成.

2) 对实数 a, $\lfloor a \rfloor$ 表示不超过 a 的最大整数. 同样，后面的记号 $\lceil a \rceil$ 表示不小于 a 的最小整数.

$k=0$ 的时候，因为 $u^0(e)$ 取值 0 或 1，所以不失一般性可以认为 $f^{*(0)} \le m$. 于是，式 (2.13) 的时间量成为 $O(m^2+mn\log n)$，是强多项式时间. 一般说来，对 N^k 得到了最小成本流 x^k 的时候，可以证明，对 N^{k+1} 的最小成本流 x^{k+1} 可以通过以 x^k 为初始解，应用算法 MINCOST，在 $O(m)$ 次迭代里得到. 因此，$k=0,1,\dots,p$ 的整个时间量成为

$$O((m^2+mn\log n)\log U), \tag{2.15}$$

是输入数据长的多项式阶.

表 2.2 最小成本流问题的算法的进展 (这里 n = 节点数，m = 边数，$C=\max_e c(e)$，$U=\max_e u(e)$. 另外，$S(n,m,C)$ 是最短路问题的时间量 (比如，$O(m+n\log n)$)，$M(n,m,U)$ 是最大流问题的时间量 (比如，$O(mn\log(n^2/m))$). 因此，表的上半部分是多项式时间算法，下半部分是强多项式时间算法.)

发现者	时间量
Edmonds, Karp (1969)[EK72]	$O(m\log U S(n,m,C))$
Röck (1980)	$O(n\log C M(n,m,U))$
Goldberg, Tarjan (1987)	$O(mn\log(n^2/m)\log(nC))$
Bertsekas, Eckstein (1987)	$O(n^3\log(nC))$
Goldberg, Tarjan (1988)[GT89]	$O(mn\log n\log(nC))$
Tardos (1985)[T85]	$O(nm^2\log^2 n+n^3m)$
Orlin (1984)	$O(m^2S(n,m))$
Galil, Tardos (1986)	$O(n^2\log nS(n,m))$
Goldberg, Tarjan (1987)	$O(nm^2\log n\log(n^2/m))$
Goldberg, Tarjan (1988)[GT89]	$O(nm^2\log^2 n)$
Orlin (1988)[O88]	$O(m\log nS(n,m))$

其后不断有对时间量的进一步改良以及强多项式时间算法的开发，其发展如表 2.2 所示．自下一节开始，介绍最初的强多项式时间的 Tardos 的算法，以及继它之后基于不同思想的 Goldberg 和 Tarjan 的算法．

2.3 最小成本循环流的最优条件

如 2.1.3 节和 2.1.4 节所述，最小成本流问题和最小成本循环流问题等价，所以，下面仅仅讨论后者．另外，把针对网络 $N=(G,c,l,u)$ 的问题模型 (2.9) 作如下变形．对 $G=(V,E)$ 的各边 $(v,w)\in E$，考虑流 $x(v,w)$ 和 $x(w,v)$．假定成立**反对称性** (antisymmetry)

$$x(w,v)=-x(v,w) \tag{2.16}$$

关于成本系数也同样假定

$$c(w,v)=-c(v,w) \tag{2.17}$$

这样一来，边 $(v,w)\in E$ 上的针对流的上下界条件 $l(v,w)\le x(v,w)\le u(v,w)$ 可以写成

$$\begin{aligned} x(v,w) &\le u(v,w) \\ x(w,v) &\le -l(v,w) \end{aligned} \tag{2.18}$$

所以，如果令 $u(w,v)=-l(v,w)$，则可以统一成仅用上界条件的表达方式．

因此，今后只要 $e=(v,u)\in E$ 就认为其反向边 $e'=(u,v)$ 也属于 E（把 E 的定义按这种方式推广），仅用上界值 u，把网络

记为 $\mathcal{N}=(G,c,u)$. 其结果、最小成本循环流问题可以写成

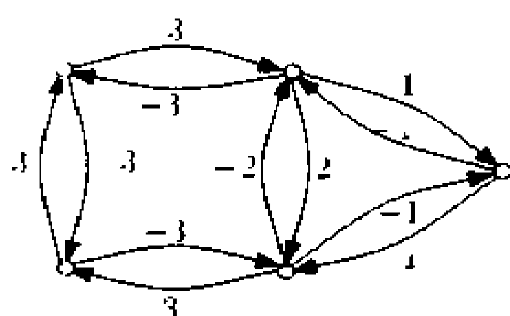
$$
\begin{array}{llll}
\text{目标函数:} & \frac{1}{2}\sum_{e\in E} c(e)x(e) & \to \text{最小} & \\
\text{约束条件:} & \sum_{e\in \mathrm{OUT}(v)} x(e)=0 & (v\in V) & (\text{流量守恒条件}) \\
 & x(e)\le u(e) & (e\in E) & (\text{容量条件}) \\
 & x(e)=-x(e') & (e\in E) & (\text{反对称性})
\end{array}
\tag{2.19}
$$

这里, e' 是 e 的反向边. 目标函数中的 1/2 对应的是, 因边的二重化, 原来的边 e 的成本

$$c(e)x(e)+c(e')x(e')=2c(e)x(e)$$

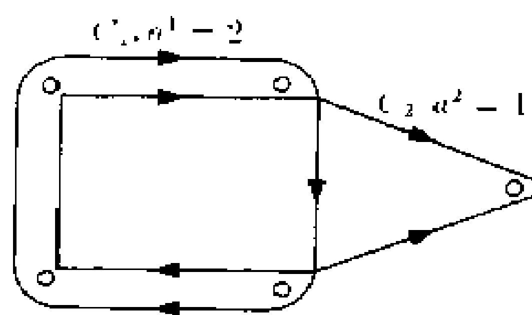
变成 2 倍. 约束条件的第一式是对从节点 v 出来的所有边求和、也即流量守恒条件.

这个新模型里面没有流 $x(e)$ 的下界条件. 满足流量守恒条件和反对称性的 x 称为 **循环流**. 满足问题 (2.19) 的所有约束条件的 x 称为 **可行流**. 可行流之中使目标函数值达最小的 x 称为 **最小成本流**. 图 2.4 (a) 给出循环流的一个例子.



(a) 循环流 x

(各边 e 上的数字表示 $x(e)$)



(b) 分解成回路 C_1

(另外还有与 C_1 反向的 C_1')

图 2.4 循环流及其分解

在说明算法之前, 先做些准备工作. 首先, 考虑任意的循

环流 x. 把 x 分解成在几个回路 C_j $(j = 1, 2, \ldots, k)$ 上循环流动的流 x^j, 注意它可以写成

$$x = \sum_{j=1}^{k} x^j \tag{2.20}$$

这里，各 x^j 可以用常数 a^j 写成

$$x^j(e) = \begin{cases} a^j, & e \in C_j\text{的情况下} \\ 0, & \text{否则} \end{cases}$$

另外，根据流的反对称性，对每个回路 C_j, 把它所包含的所有边反向后得到的回路 $C_{j'}$ 也有流循环，并成立 $a^{j'} = -a^j$. 图 2.4(b) 是把同图 (a) 的循环流分解成回路 C_j 后的结果. 这里，为简单起见，省略了各 C_j 的所有边反向后得到的回路 $C_{j'}$.

另外, 即使在模型 (2.19) 里, 也可以针对可行流 x 推广 2.2.3 的余网络，定义为 $N_x = (G_x, c, u')$, 这里，图 $G_x = (V, E_x)$ 的边集合 E_x 由

$$e \in E_x \Longleftrightarrow e \in E \text{ 而且 } u(e) - x(e) > 0 \tag{2.21}$$

决定，对各 $e \in E_x$ 有

$$u'(e) = u(e) - x(e)$$

下面导入**势函数** (potential function) $p : V \to R$ 来确定边 $(v, w) \in E$ 的**相对成本系数** (参照 1.3 节针对一般线性规划的定义),

$$\bar{c}(v, w) = c(c, w) + p(v) - p(w) \tag{2.22}$$

$p(v)$ 可以解释为线性规划问题 (2.19) 的对偶问题 (1.7 节) 里的对偶变量. 另外，由

$$\bar{c}(w, v) = c(w, v) + p(w) - p(v) = -c(v, w) + p(w) - p(v) = -c(v, w)$$

知道，$\bar{c}$ 也有反对称性. 进一步，因沿着一个回路 C 取成本和就有

$$\sum_{e\in C} c(e) = \sum_{e\in C} \bar{c}(e)$$

所以，考虑 (2.20)，得到下面的引理.

引理 2.3 (相对成本) 最小成本循环流问题 (2.19) 里，成本系数 c 变换为 $\bar{c}$ 后，最优解及其值无变化.

进一步，下面的引理刻划了最小成本流的特征，成为以下强多项式算法的出发点.

引理 2.4 (最优条件) 对于最小成本循环流问题 (2.19) 的可行流 x，以下三条件等价.

(i) x 是最小成本流.

(ii) 余网络 N_x 没有成本和为负的回路.

(iii) 对所有的边 $e \in E$，存在有满足条件

$$u(e) - x(e) > 0 \Rightarrow \bar{c}(e) \geq 0 \tag{2.23}$$

的势 p (这个条件的前半部分意味着 e 是 N_x 的边. 另外，如果 $e = (v, w)$ 和 $e' = (w, v)$ 两者都是 N_x 的边，则根据这个条件成立有 $\bar{c}(e) = \bar{c}(e') = 0$.)

证明 (i)⇒ (ii). 在 N_x 中存在成本和为负的回路 C 的情况下，利用

$$\Delta = \min\{u(e) - x(e) | e \in C\} \quad (> 0) \tag{2.24}$$

把 x 修正为

$$x'(e) = \begin{cases} x(e) + \Delta, & e \in C\text{的情况下} \\ x(e) - \Delta, & (e\text{的反向边})e' \in C\text{的情况下} \\ x(e), & \text{其它情况下} \end{cases} \tag{2.25}$$

很容易知道，x' 也是可行流，而且其成本比 x 更小．与 x 是最小成本流矛盾．

(ii) $\to$ (iii) N_x 中把 c 看成边长，任意选取的一个节点 $s \in V$，求出它到所有其它节点 v 的最短路，令 $p(v)$ 为其长度 (N_x 没有成本和为负的回路，因而可以断言 $p(v)$ 的存在性) 这个时候对任意的 $(v, w) \in E_x$，

$$\bar{c}(v, w) = c(v, w) + p(v) - p(w) \geq 0$$

这是因为，如果 $p(v) + c(v, w) < p(w)$ 的话，从 s 向 v 前进长度 $p(v)$，其后通过边 (v, w) 到达 w 就得到比 $p(w)$ 更短的路，与 $p(w)$ 的定义相矛盾．

(iii) $\Rightarrow$ (i)．令 x^* 为最小成本流，而且设 $x \neq x^*$．考虑 $\bar{x} = x^* - x$，很容易知道 $\bar{x}$ 是循环流，因而可像式 (2.20) 那样分解成几个回路 C_j 上的流．满足 $a^j > 0$ 的回路 C_j 的边 e 上，根据 $x(e) < x^* \leq u(e)$，由条件 (2.23) 有 $\bar{c}(e) \geq 0$，C_j 的全部边的成本加起来满足

$$\sum_{e \in C_j} \bar{c}(e) a^j = \sum_{e \in C_j} c(e) a^j \geq 0$$

在 C_j 的所有边反向后得到的回路 $C_{j'}$ 上，尽管根据 $\bar{x}$ 的反对称性成立有 $a^{j'} = -a^j < 0$，但是同时也考虑到 $\bar{c}$ 的反对称性的话就知道上式仍然成立．结果，

$$c^T x^* - c^T x + c^T x \geq c^T x$$

再根据 x^* 的最优性 $c^T x^* = c^T x$，因而 x 也是最小成本流．

取这个引理的条件 (iii) 之对偶，可以描述如下．

$$\bar{c}(e) < 0 \Rightarrow x(e) = u(e) \tag{2.26}$$

另一方面，如果 $c(e) \geq 0$，除 $x(e) \leq u(e)$ 外关于 $x(e)$ 无其它条件．但是，有必要注意在 $\bar{c}(e) > 0$ 的时候，关于 e 的反向边 e' 成立 $\bar{c}(e')(= -c(e)) < 0$，所以根据式 (2.26) 有 $x(e') = u(e')$，也就是根据反对称性成立

$$\bar{c}(e) > 0 \Rightarrow x(e) = -u(e') \tag{2.27}$$

2.4 Tardos 的强多项式时间算法

考虑网络 $N = (G, c, u)$ 的最小成本循环流问题 (2.19)．为描述算法的方便，导入具有下述性质的边集合 $T \subset E$：对各 $(v, w) \in T$，最小成本流 x 满足

$$x(v, w) = u(v, w) \text{ 或者 } x(v, w) = -u(w, v) \tag{2.28}$$

(并且知道其中到底是哪个成立)．算法最先从 $T := \emptyset$ 出发，迭代中逐渐扩大 T．当 $T = E$ 的时候，所有边 e 的 $x(e)$ 被固定了．这就意味着求出了最小成本流 x，可以结束计算．另外，根据流的反对称性，如果 $(v, w) \in T$，则可以假定 $(w, v) \in T$．

为使得上述 T 的扩大成为可能，Tardos 的算法构造具有下列性质的 N 的近似网络 $N' = (G, c', u)$．

- 针对 N' 的最小成本循环流问题在强多项式时间内可解．
- 其结果，至少找到一条属于 T 的新边．

这样一来，最多进行 m 次构造并解出 N' 的步骤就成立有 $T = E$，所以必要的计算量是强多项式时间．

由类似 2.2.3 的度量法的思想来构造近似网络 N'，其基础是放松最优条件 (2.23) 后的 ε-**最优**(ε-optimal) 条件：对任意的 $e \in E$，可行流 x 和势 p 满足

$$u(e) - x(e) > 0 \Rightarrow \bar{c}(e) \geq -\varepsilon \tag{2.29}$$

这里，$\varepsilon \geq 0$ 是给定的常数．原来的条件 (2.23) 对应于 $\varepsilon = 0$ 的情况．另外，条件 (2.29) 也可以换成

$$\bar{c}(e) < -\varepsilon \Rightarrow x(e) = u(e) \tag{2.30}$$

引理 2.5 (固定流的条件) 对网络 $N = (G, c, u)$ 考虑 ε-最优的可行流 x 和势 p. 对某条边 $e^* \in E$，如果成立

$$\bar{c}(e^*) \leq -n\varepsilon \tag{2.31}$$

(此时，根据条件 (2.30)，$x(e^*) = u(e^*)$)，则任意的最小成本流 x^* 满足

$$x^*(e^*) = u(e^*)$$

(也即，对 e^* 及其反向边 e'' 可以认定 $e^*, e'' \in T$)．这里，n 是 G 的节点数，另外还假定 $n \geq 2$.

证明 考虑满足 $x'(e^*) < x(e^*)$ 的可行流 x'. 根据反对称性，e^* 的反向边 e'' 成立 $x'(e'') > x(e'')$ 以及 $\bar{c}(e'') \geq n\varepsilon$. 因此，如果考虑循环流 $x' - x$，则根据 $x'(e'') - x(e'') > 0$ 和流量守恒条件，存在包含边 e'' 的回路 C 满足

$$x'(e) - x(e) > 0 \quad (e \in C) \tag{2.32}$$

下面利用

$$\Delta = \min\{x'(e) - x(e) | e \in C\} \quad (> 0)$$

把 x' 修正为

$$x''(e) = \begin{cases} x'(e) - \Delta, & e \in C \\ x'(e) + \Delta, & (e\text{的反向边})e' \in C \\ x'(e), & \text{其它情形} \end{cases}$$

根据定义，x'' 也是可行流，其成本是

$$\begin{aligned} \sum_{e\in E} c(e)x''(e) &= \sum_{e\in E} c(e)x'(e) - \Delta\sum_{e\in C} c(e) + \Delta\sum_{e'\in C} c(e) \\ &\le \sum_{e\in E} c(e)x'(e) - 2\Delta(c(e'') - (|C| - 1)\varepsilon) \\ &\quad (\text{根据式}(2.29)\text{和反对称性}) \\ &\le \sum_{e\in E} c(e)x'(e) - 2\Delta(n\varepsilon - (n-1)\varepsilon) < \sum_{e\in E} c(e)x'(e) \end{aligned}$$

这里 $|C|$ 是 C 所包含的边数．从这个结果可以知道，x' 不给出最小成本．

因此下面考虑如何计算满足这个引理条件的 x 和 p. 假设给定了满足式 (2.28) 的某个边集合 T, 引进满足流量守恒条件和 $y(e)=0$ $(e \in T)$ 的 R^m 的子空间

$$Q = \{y \in R^m \mid \sum_{(v,w)\in E-T} y(v,w) = 0\ (v \in V)\ \text{而且}\ y(e) = 0\ (e \in T)\}$$

把成本向量 c 投影到 Q 并记之为 c^*. 这个 c^* 可以通过解联立方程组

$$\sum_{(v,w)\in E-T} (c(v,w) + p^*(v) - p^*(w)) = 0\ (v \in V) \tag{2.33}$$

在得到 p^* 之后令

$$c^*(v,w) = \begin{cases} c(v,w) + p^*(v) - p^*(w), & (v,w) \in E - T \\ 0, & (v,w) \in T \end{cases} \tag{2.34}$$

而求出．这个时候如果 $c^*(e)=0\ (e \in E)$，则根据引理 2.3，对集合 T 的边满足式 (2.28) 的任意可行流都是最优解．后述算法根据这个条件来结束计算．

反之，如果存在某个 $e \in E$ 使得 $c^*(e) \neq 0$，则用适当的正数 L 乘以 c^* 后加以正规化使之满足

$$\begin{aligned} c'' &= Lc^* \\ c''_{\max} &= n\sqrt{m} \end{aligned} \tag{2.35}$$

这里，一般对以 $d(e)\ (e \in E)$ 为分量的向量 d，采用记号

$$d_{\max} = \max\{|d(e)| \mid e \in E\}$$

即使把上面的 c'' 作为成本向量，引理 2.4 的最优条件也不会变．之后，把 c'' 变为整数

$$c'(e) = \lceil c''(e) \rceil \quad (e \in E) \tag{2.36}$$

由此得到近似成本向量 c'．

现在，在得到的近似网络 $N' = (G, c', u)$ 里，固定 T 所指定的变量后，使用适当的算法，和最小成本循环流 x 一起求出满足最优条件的 p．利用这个 p，令

$$\bar{c}''(v,w) = c''(v,w) + p(v) - p(w) \tag{2.37}$$

则上面的 x 和 p 在 $\varepsilon = 1$ 的情况下满足 ε- 最优条件 (2.30)．其理由是，如果 $\bar{c}''(v,w) < -1$，则

$$\begin{aligned} \bar{c}'(v,w) &= c'(v,w) + p(v) - p(w) \\ &= c''(v,w) + p(v) - p(w) + (c'(v,w) - c''(v,w)) \\ &< -1 + 1 = 0 \end{aligned}$$

根据 N' 里最优条件 (2.26)，这也就得出 $x(v,w)=u(v,w)$

进一步成立下面的引理.

引理 2.6 (满足固定条件的边的存在性) 在上述的 N', T, x, p, c'', $\bar{c}''$ 里，对某个 $e^* \in E-T$ 成立

$$c''(e^*) \leq -n \tag{2.38}$$

证明 由式 (2.37) 的 $\bar{c}''$ 按如下方式构造 $\hat{c}$.

$$\hat{c}(e)=\begin{cases}\bar{c}''(e), & e\in E-T\\ 0, & e\in T\end{cases}$$

向量 $\hat{c}-c''$ 和 c'' 正交 (也就是，$\sum_{e\in E}(\hat{c}(e)-c''(e))c''(e)=0$) 其原因是，当 $e\in T$ 时，根据定义 $c''(e)$, $\hat{c}(e)$ 同时为 0，当 $e\in E-T$ 时，取和得到

$$\begin{aligned}&\sum_{e\in E-T}(\hat{c}(e)-c''(e))c''(e)\\ &=\sum_{(v,w)\in E-T}(p(v)-p(w))c''(v,w) \text{ (根据式(2.37))}\\ &=2\sum_{v\in V}(p(v)\sum_{(v,w)\in E-T}c''(v,w))=0\\ &\text{(根据式(2.33)} \sim \text{(2.35)以及}c''\text{的反对称性.)}\end{aligned}$$

依据向量 $\hat{c}-c''$ 和 c'' 的正交性，可以推出 $\|c''\|\leq\|\hat{c}\|$. 这里 $\|\cdot\|$ 是欧几里得范数

$$\|c\|=(\sum_{e\in E}c^2(e))^{1/2}$$

这样一来，

$$\begin{aligned}\hat{c}_{\max} &\geq \|\hat{c}\|/\sqrt{m} \text{ (根据范数的性质)}\\ &\geq \|c''\|/\sqrt{m}\geq c''_{\max}/\sqrt{m}=n \text{ (根据式(2.35))}\end{aligned}$$

也即，对于某个 $(w,v)\in E-T$，有 $\bar{c}(w,v)-\bar{c}''(w,v)\geq n$．根据 $\bar{c}''$ 的反对称性，该式意味着对某个 $(v,w)\in E-T$ 有 $\bar{c}''(v,w)\leq -n$．

把引理 2.5 用到这个结果上（注意到 $\epsilon=1$），可以新固定 $x(e^*)=u(e^*)$ 以及 $x(e'')=-u(e^*)$（e'' 是 e^* 的反向边），把 e^* 和 e'' 加到 T 里．下面的算法是进行这种迭代的程序．

算法 TARDOS (Tardos 的强多项式算法)

输入： 网络 $N=(G,c,u)$．

输出： N 的最小成本流 x．

第一步（初始化）： $T:=\emptyset$，$x:=$（任意的可行流）（判定 N 的可行性以及求出可行流可以像 2.1.4 所述的那样，作为最大流问题可在多项式时间内完成．）

第二步（修正成本向量）： 根据式 (2.33)～(2.36) 修正 c，得出 c^*，c'' 和 c'．如果 $c^*(e)=0$ $(e\in E)$，则结束计算；现在的 x 给出最小成本流．否则，对网络 $N'=(G,c',u)$ 在 T 的条件 (2.28)，(2.19) 下解最小成本循环流问题，令得到的流为 x．

第三步（更新 T）： 对满足引理 2.6 的条件 $\bar{c}''(v,w)\leq -n$ 的所有边 (v,w)，固定 $x(v,w)=u(v,w)$，$x(w,v)=-u(v,w)$．把边 (v,w) 和 (w,v) 加入 T，回到第二步．

下面估计上述算法的时间量．各次迭代中主要的计算是在第二步．为把成本向量 c 变换成向量 c'，首先必须解联立方程组 (2.33)．因为有 n 个变量和 n 个条件，所以时间 $O(n^3)$ 是足够的．其次，对网络 N'，在关于 T 的约束下求出最小成本流 x．像 2.1.4 所述的那样，最小成本循环流问题和最小成本流问题等价，同时考虑到 $c'_{\max}=O(n\sqrt{m})$（式 (2.35) 和 (2.36)），如果采

用表 2.2 的第五个最小成本流算法 (Goldberg 和 Tarjan), 则所要的时间是

$$O(mn\log n\log(nc'_{\max})) = O(mn\log n\log n^2m^{1/2}) = O(mn\log^2 n)$$

除此之外的计算显然在 $O(m+n)$ 的时间内可以完成. 第二步和第三步的每次迭代中 T 严格增大, 所以最多有 m 次. 因而, 整个时间量可以估计为

$$O(nm^2\log^2 n + n^3m) \tag{2.39}$$

这是强多项式时间

定理 2.1 (TARDOS 的强多项式性) 算法 TARDOS 在强多项式时间内正确解出最小成本循环流问题.

2.5 Goldberg 和 Tarjan 的强多项式时间算法

这里介绍针对最小成本循环流问题 (2.19) 的另一个强多项式时间算法, Goldberg 和 Tarjan 的成果 [GT 89]. 基于引理 2.4 的条件 (ii), 只要余网络 N_x 中存在有成本和为负的回路 C, 按式 (2.24), (2.25) 对沿着 C 的流进行量仅为 Δ 的修正 (称之为 C 的**删除操作** (cancel)). 这个算法就是反复这种删除操作的方法. 其特点是, 用到的 C 取为使 (沿着 C 的) 平均成本

$$\mu_C = \sum_{e\in C} c(e)/|C| \tag{2.40}$$

达到最小的回路. 这里, $|C|$ 表示 C 的边数.

算法 GT (Goldberg 和 Tarjan 的强多项式时间算法)

输入：网络 $N=(G,c,u)$.

输出：N 的最小成本流 x.

第一步 (初始化)：$x :=$ (任意的可行流). (参照算法 TARDOS (2.4 节) 中第一步的评论.)

第二步 (余网络)：构造余网络 $N_x=(G_x=(V,E_x),c,u)$. 如果 N_x 中没有成本和为负的回路则结束计算. 如果有这样的回路则进入第三步.

第三步 (回路的删除)：求出使式 (2.40) 的平均成本 μ_C 达最小的回路 C, 按式 (2.24), (2.25) 进行 C 的删除操作. 把得到的流 x' 作为 x 重新回到第二步.

下面利用几个引理证明算法 GT 的迭代次数可以控制在强多项式时间内，其结果，GT 成为强多项式时间算法.

前节式 (2.29) 的 ε- 最优性 (这里，$\varepsilon>0$) 是关于可行流 x 和势 p 的. 如果对于 x, 存在某个势 p, 使得对任意的 $e\in E$ 成立

$$u(e)-x(e)>0\ (\text{也即}, e\in E_x)\Rightarrow \overline{c}(e)>-\varepsilon \qquad (2.41)$$

则称 x 为 ε- 最优. 这里，E_x 是余网络 N_x 的边集合.

引理 2.7 (ε- 最优流的最优性质) 假定网络 $N=(G,c,u)$ 的成本系数 $c(e)$ 全部是整数. 这个时候，如果可行流 x 对某个 $\varepsilon<1/n$ 为 ε- 最优，则 x 是最小成本流. 这里，n 是 G 的节点数.

证明 令 C 为余网络 N_x 的任意的回路. 根据 x 的 ε- 最优性质 (注意 N_x 的所有边 e 成立 $u(e)-x(e)>0$), 沿着 C 的成本满足

$$\sum_{e\in C}c(e)=\sum_{e\in C}\overline{c}(e)\geq-|C|\varepsilon\geq-n\varepsilon>-1$$

因为 $\sum c(e)$ 是整数，也就有

$$\sum_{e \in C} c(e) \geq 0$$

故 N_x 没有成本和为负的回路．根据引理 2.4 (ii) x 是最优的．

下面，对可行流 x，把使 x 成为 ε- 最优的最小 $\varepsilon(>0)$ 记为 $\varepsilon(x)$．另外，在余网络 N_x 里，回路的平均成本 μ_C 的最小值记为 $\mu(x)$ 两者之间成立下面的有趣关系．

引理 2.8 (ε 和 μ 的关系) 可行流 x 不是最优的时候，成立有 $\varepsilon(x) = -\mu(x) (>0)$

证明 设实现 $\varepsilon(x)$ 的势设为 p，根据定义，p 是使

$$\min_{(v,w) \in E_x} (c(v,w) + p(v) - p(w))$$

达最大者．求这种 p 的问题可以写成下面的线性规划问题，得到的最优值是 $-\varepsilon(x)$

$$\begin{aligned} &\text{目标函数：} \quad \varepsilon \longrightarrow \text{最大} \\ &\text{约束条件：} -\varepsilon \leq c(v,w) + p(v) - p(w) \quad ((v,w) \in E_x) \end{aligned} \tag{2.42}$$

该问题的对偶问题 (参照 1.7 节) 是

$$\begin{aligned} &\text{目标函数：} \sum_{e \in E_x} c(e)x(e) \longrightarrow \text{最小} \\ &\text{约束条件：} \sum_{e \in \mathrm{OUT}_x(v)} x(e) - \sum_{e \in \mathrm{IN}_x(v)} x(e) = 0 \quad (v \in V) \\ &\qquad \sum_{e \in E_x} x(e) = 1 \\ &\qquad x(e) \geq 0 \quad (e \in E_x) \end{aligned} \tag{2.43}$$

这里，$\mathrm{OUT}_x(v)$ 和 $\mathrm{IN}_x(v)$ 分别表示 N_x 里流出 v 和流入 v 的边集．如果不考虑约束条件 $\sum x(e)=1$，则该问题成为 N_x 的在式 (2.9) 意义下的最小成本循环流问题之特别情形．于是，像式 (2.20) 所述的那样，问题 (2.43) 的可行解可以分解成在几个回路 C_j 上循环流动的流 x^j．但是，问题 (2.43) 里，没有给出各边的流 $x(e)$ 的上界值，所以，不失一般性可以认为最优解是仅在一个回路 C 上循环流动的流．根据约束条件

$$\sum_{e\in C} x(e)=1$$

这种流 x 成为

$$x(e)=\begin{cases} 1/|C|, & e\in C\text{的情况下} \\ 0, & e\notin C\text{的情况下} \end{cases}$$

目标函数满足条件

$$\sum_{e\in E_x} c(e)x(e)=\sum_{e\in C} c(e)x(e)=\sum_{e\in C} c(e)/|C|=\mu_C$$

也就是，C 是使平均成本 μ_C 达最小的回路．其结果，根据线性规划问题的对偶定理 (定理 1.5)，主问题 (2.42) 的最优值与对偶问题 (2.43) 的最优值一致．也即，$\varepsilon(x)=-\mu(x)$．

另外，因为使 μ_C 达最小的回路 C 满足如下两条件

$$\mu(x)=\sum_{e\in C} c(e)/|C|=\sum_{e\in C} \bar{c}(e)/|C|=-\varepsilon(x)$$

$$c(e)\leq \ \varepsilon(x)\quad (e\in C)\quad (\text{问题 (2.42) 的约束条件})，$$

把它们结合起来就知道有

$$\bar{c}(e)=-\varepsilon(x)\quad (e\in C) \tag{2.44}$$

算法 GT 在每次迭代中求出 $\mu(x) = -\varepsilon(x)$ (引理 2.8). 根据下面两个引理知道这个 $\varepsilon(x)$ 以某种速度减小下去.

引理 2.9 ($\varepsilon(x)$ 的非增加性) 算法 GT 在第二步进行删除操作后, $\varepsilon(x)$ 是非增加的.

证明 在进行删除操作之前的时刻, 实现 $\varepsilon(x)$ 的势记为 p. 也即, 对所有的 $(v, w) \in E_x$,

$$\bar{c}(v,w) = c(v,w) + p(v) - p(w) \geq -\varepsilon(x) \tag{2.45}$$

把对 C 进行删除操作之后得到的流记为 x', C 至少有一条边达到饱和因而不属于 $E_{x'}$. 与此同时, 仅仅限于把 C 中边 (v, w) 反向之后的边 (w, v) 作为 $E_{x'}$ 中新入边 (一般有多条). 在 C 上因成立式 (2.44), 故对于这种边 (w, v), 根据反对称性,

$$\bar{c}(w,v)(= c(w,v) + p(w) - p(v)) = \varepsilon(x)\ (> 0)$$

于是, 对于所有的 $(v', w') \in E_{x'}$, 式 (2.45) 成立. 这就意味着 $\varepsilon(x') \leq \varepsilon(x)$. 再把这个 x' 重新看成 x 就知道引理成立.

引理 2.10 ($\varepsilon(x)$ 的减少率) 算法 GT 里, 第二步的删除操作进行 m 次后, 成立

$$\varepsilon(x') \leq (1 - 1/n)\varepsilon(x)$$

这里, x 是迭代前的流, x' 是 m 次迭代后的流. 另外, n 表示网络 N 的节点数, m 是边数.

证明 利用迭代前实现 $\varepsilon(x)$ 的势 p, 记

$$\bar{c}(v,w) = c(v,w) + p(v) - p(w)$$

几次迭代后, 流要变化, 各次迭代中使平均成本最小的回路 C 里条件

$$\bar{c}(e) < 0 \quad (e \in C)$$

并不恒成立. 为此, 考虑下面两种情况.

(a) m 次迭代中, 使平均成本达最小的回路 C 的边 e 全满足 $\bar{c}(e) < 0$. 这种情况下, 像引理 2.9 所述的那样, 每次迭代中满足 $\bar{c}(e) < 0$ 的边至少有 1 条被排出余网络. 另外, 新加入的边全满足 $\bar{c}(e) > 0$, 所以 m 次迭代后对所有的边 $e \in E_{x'}$ 成立

$$\bar{c}(e) > 0$$

也就是 $\varepsilon(x') = 0$, 引理成立.

(b) 在几次迭代后, 某条边 $e^* \in C$ 满足 $\bar{c}(e^*) \geq 0$. 考虑最初出现这种情况的时刻, 记得到的流为 x''. 因 (根据引理 2.9 的证明) 所有的边 $e \in C$ 满足 $\bar{c}(e) \geq -\varepsilon(x)$, 其平均成本是

$$\begin{aligned}\sum_{e \in C} \bar{c}(e)/|C| (&= -\varepsilon(x''), \text{根据引理 } 2.8)\\ &\geq -(1 - 1/|C|)\varepsilon(x) \geq -(1 - 1/n)\varepsilon(x)\end{aligned}$$

也就是, $\varepsilon(x') \leq \varepsilon(x'') \leq (1 - 1/n)\varepsilon(x)$.

为把这些结果与算法 GT 的强多项式性联系起来, 引进一个新定义. 称边 e 是 ε-**固定的** (ε-fixed), 如果在任意的 ε- 最优的可行流 x 里, $x(e)$ 取同一值. 特别是, 对最优流 x^* 也成立 $x(e) = x^*(e)$. 下面的引理推广了前节的引理 2.5.

引理 2.11 (ε- 固定的充分条件) 对某个 $\varepsilon > 0$, 设可行流 x 和势 p 是 ε- 最优的. 这个时候, 如果对边 $e^* \in E$ 成立

$$\bar{c}(e^*) \leq -2n\varepsilon$$

则 e^* 是 ε- 固定的.

证明 根据 ε- 最优的定义 (2.29) 或者式 (2.30), 边 e^* 上成立 $x(e^*) = u(e^*)$. 和引理 2.5 的证明一样, 考虑满足 $x'(e^*) < x(e^*)$

的任意可行流 x'，证明 x' 不是 ε 最优的．像引理 2.5 的证明那样，取包含 e^* 的反向边 e''、满足式 (2.32) 的回路 C，有

$$\begin{aligned}\sum_{e\in C} c(e) &> c(e'') - (|C|-1)\varepsilon \quad (x\text{和}p\text{的}\ \varepsilon\ \text{最优性})\\ &> 2n\varepsilon - (n-1)\varepsilon \quad (c(e^*)\text{的条件和反对称性})\\ &> n\varepsilon\end{aligned}$$

因此，如果考虑 C 的反向后的回路 C'，(根据式 (2.32)) C' 的所有边全属于余网络 $N_{x'}$，特别是 $e^* \in C'$．而且，C' 的平均成本满足

$$\sum_{e'\in C'} \bar{c}(e')/|C'| = -\sum_{e\in C} \bar{c}(e)/|C| < -n\varepsilon/|C| < -\varepsilon$$

这表示 x' 不是 ε- 最优的．

如果在算法 GT 的某个时刻，边 e 是 $\varepsilon(x)$- 固定的，则在其后即使 x 变化，因 $\varepsilon(x)$ 的非增加性 (引理 2.9)，边 e 恒为 $\varepsilon(x)$- 固定的．因此，如果所有的边都是 $\varepsilon(x)$- 固定的，则可以把这个时候的 x 作为最小成本流输出，结束计算．下面的引理告诉我们，到这一步所需要的迭代次数是强多项式阶．这里，证明中用到了对 $n \ge 2$ 的下述不等式．ln 是自然对数．

$$(1-1/n)^{n(\ln n+1)} < 1/2n$$

引理 2.12 (GT 的迭代次数) 算法 GT 的第二步和第三步在 $O(nm^2 \log n)$ 次迭代之后结束计算．

证明 令 $k = nm\lceil \ln n + 1\rceil$．证明在连续 k 次迭代后，至少有一条新边成为 $\varepsilon(x)$- 固定的．记这样的 k 次迭代开始时的流为 x，结束时的流是 x'．根据引理 2.10，

$$\varepsilon(x') \le (1-1/n)^{n\lceil \ln n+1\rceil}\,\varepsilon(x) \le \varepsilon(x)/2n$$

记第一次迭代所选取的回路为 C，根据 (2.44)，任意的 $e \in C$ 满足

$$\bar{c}(e) = -\varepsilon(x) < -2n\varepsilon(x')$$

因而根据引理 2.11，在 k 次迭代结束的时候，e 是 $\varepsilon(x')$- 固定的．而且，第一次迭代中 $x(e)$ 被修正过，所以 $e \in C$ 在当时还未被 $\varepsilon(x)$- 固定．结果，这种 k 次迭代最多经过 m 次重复后，m 条边全被 $\varepsilon(x)$- 固定，计算可以结束．于是，迭代的次数最多是 $mk = O(nm^2 \log n)$．

为估计算法 GT 的整体计算量，除迭代次数之外，还必须知道各步骤所要的时间．余网络 N_x 的构造和回路的删除操作显然在 $O(m)$ 时间内可能完成 (如果下点工夫，还可以更快些)，时间量上重要的是在 N_x 中找出使平均成本最小的回路的问题 (其结果，也可以判定 N_x 中是否存在成本和为负的回路)．以下介绍解这个问题的一个算法，并证明它可在 $O(mn)$ 时间内完成．

为简单起见，假定 N_x 强连通 (即，对任意的 2 节点 v, w，存在有通过它们的回路)．如果不是强连通，则只要对每个强连通部分分别利用以下的算法即可．任意选取 N_x 的一个节点 $s \in V$，对非负整数 k 和节点 v，定义

$F_k(v) =$ (正好使用 k 条边从 s 到达 v 的路的成本之最小值)

这里的路允许两次以上通过同一顶点．如果使用 k 条边无法从 s 到 v，则 $F_k(v) = \infty$

引理 2.13 ($\mu(x)$ 的计算法) 余网络 N_x 里，平均成本最小的回路的值 $\mu(x)$ 由

$$\mu(x) = \min_{v \in V} \max_{0 \le k \le n-1} [(F_n(v) - F_k(v))/(n-k)] \qquad (2.46)$$

给出.

证明 先假定 $\mu(x)=0$，在这种情况下要证明上式右边等于 0. $\mu(x)=0$ 的情况下不存在成本和为负的回路，因而，存在 s 到各 $v\in V$ 的最小成本的路. 记这种最小成本值为 $c^*(v)$. 另外，最小成本路上不必两次以上通过同一节点，所以，

$$c^*(v)=\min_{0\le k\le n-1}F_k(v)$$
$$F_n(v)\ge c^*(v),$$

于是，

$$F_n(v)-c^*(v)=\max_{0\le k\le n-1}[F_n(v)-F_k(v)]\ge 0$$

也就得到

$$\max_{0\le k\le n-1}[(F_n(v)-F_k(v))/(n-k)]\ge 0$$

为证明这个式子是等式，即

$$\max_{0\le k\le n-1}[(F_n(v)-F_k(v))/(n-k)]=0 \tag{2.47}$$

只要证明存在满足 $F_n(v)=c^*(v)$ 的 $v\in V$. 根据假定 $\mu(x)=0$，存在有成本和为 0 (即平均成本为 0) 的回路 C. 像图 2.5 那样，设 w 为 C 的一个节点，$P(w)$ 是具有成本值 $c^*(w)$ 的从 s 到 w 的路，则继 $P(w)$ 之后沿 C 环绕几周后到 w 的路 $P'(w)$ 也是到达 w 的最小成本路. 可以假设 $P'(w)$ 的边数在 n 以上. 因此，如果 $P'(w)$ 里从 s 出发经过 n 条边到达 w'，则 $F_n(w')=c^*(w')$ (如果 $F_n(w')>c^*(w')$，把 $P'(w)$ 的最初的 n 条边形成的部分用到 w' 的最小成本路替换后，得出到达 w 的具有更小成本的路，导致矛盾.) 把这个 w' 看成上面的 v 即可.

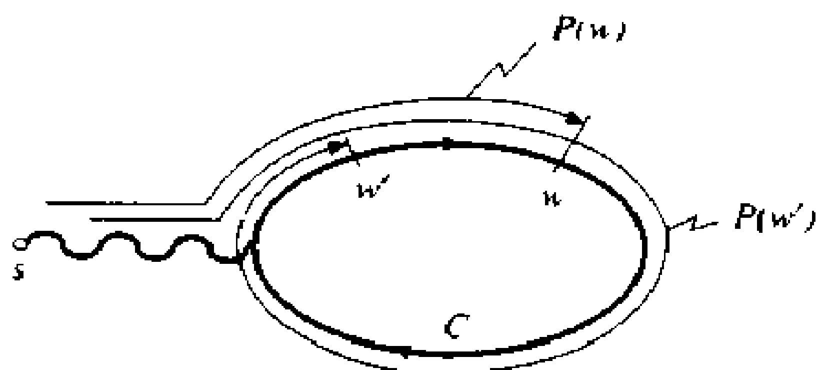


图 2.5 引理 2.13 的证明

其次，为考虑 $\mu(x)\neq 0$ 的情况，利用常数 a 把成本系数 $c(e)$ 替换为

$$c'(e)=c(e)-a\quad(e\in E)$$

其结果，$\mu(x)$ 仅减少 a, $F_n(v)$ 减少 na, $F_k(v)$ 减少 ka, $(F_n(v)-F_k(v))/(n-k)$ 减少 a, 也即式 (2.46) 的两边都减少 a, 所以如果关于成本系数 $c'(e)$ 式 (2.46) 成立的话、对原来的成本系数式 (2.46) 的证明也就完成了．因此，如果利用 $a=\mu(x)$ 修正成本系数，则新的成本系数显然成立 $\mu(x)=0$, 所以式 (2.47), 也即式 (2.46) 成立．

$F_k(v)$ 的计算可以从

$$F_0(s)=0,\quad F_0(v)=\infty\quad(v\in V-\{s\})\tag{2.48}$$

开始，通过递推式

$$F_k(v)=\min_{(w,v)\in E_x}(F_{k-1}(w)+c(w,v))\tag{2.49}$$

以 $k=1,2,\cdots,n$ 的顺序解出 (这是从所谓动态规划的思想得到的结果). 必要的时间量是 $O(mn)$. 一但求出所有的 $F_k(v)$, 式 (2.46) 的右边可以在 $O(n^2)(\leq O(mn))$ 时间内直接计算出．为具

体求具有平均成本 $\mu(x)$ 的回路 C, 对实现式 (2.46) 的 v 和 k, 求出从 s 到 v 的由 n 条边组成的最小成本路 $P(v)$ (只要记忆计算式 (2.49) 时实现最小值的边 (w,v) 即可构成) 输出其后半的 $n-k$ 条边形成的回路 (沿着引理 2.13 的证明思路可以知道这是回路) 即可. 为此需要的时间量是 $O(n)$

概括上面的讨论, 可以得到下面的定理

定理 2.2 (GT 的强多项式性) 算法 GT 正确解出最小循环流问题 (2.19), 迭代次数是 $O(nm^2\log n)$, 时间量是 $O(n^2m^3\log n)$, 也即强多项式时间.

为进一步减少时间量, Goldberg 和 Tarjan [GT 89] 考察了有组织地引进利用容易计算的回路, 减少最小平均成本回路计算次数的算法. 其结果, 成功地使算法具有精炼的数据结构并使时间量减少到 $O(nm^2\log^2 n)$. 像 2.2 节中表 2.2 所示的那样, 这是到现在为止知道的最快的算法之一.

2.6 文献及其他话题

作为网络最优化问题, 本章叙述了最短路问题、最大流问题, 最小成本流问题 (以及与之等价的最小成本循环流问题). 对除此之外的匹配问题, 最小树问题等各种问题也都有广泛研究. 下章讨论的旅行商问题也是有代表性的网络最优化问题. 讨论这些各种各样问题的书已经出版了相当多. 这里, 除列举经典的 Ford Fulkerson [FF 62] 外, 还加上伊理, 藤重, 大山 [IFO 86], Tarjan [T83], 进一步, Ahuja, Magnanti, Orlin [AMO 93] 等. [T83] 详细介绍了算法高速化时有用的数据结构. [AMO 93] 在了解关于这些话题的最近成果的广泛应用方面很有用.

2.2.1 的针对最短路问题的戴克斯特拉法是 Dijkstra [D59] 在

1959 提出的. 文献 [IF91] 给出了具体算法 另有其它最短路的高速算法, 比如 Imai 和 Iri [II 84].

最大流问题也是这个领域的经典话题, 但是, 像表 2.1 所示的那样, 最近算法有显著的进步. 2.2.2 给出 Edmonds 和 Karp [EK 72] 的强多项式时间算法的概要, 其它算法的讲解在上述的 [T83] [AMO 93] 或者岩野 [I 89] 等里有.

关于最小成本流问题, 以 2.4 节 Tardos [T 85] 以及 2.5 节 Goldberg 和 Tarjan [GT 89] 的算法为中心, 稍微详细地介绍了强多项式时间算法. 成为其基础的 2.3 节的最优条件等来源于线性规划的对偶理论, 这已在 Ford 和 Fulkerson [FF 62] 里作了详细讨论. Tardos 的算法的解说在永持 [N 90] 里也有. 除此之外, 表 2.2 中, Tardos 的算法的对偶版本, Fujishige [F86], 以及时间量最少的算法之一 Orlin 算法 [O 88] 也是值得注意的. 另外, 表 2.1 和表 2.2 中的年份所表示的是最初发表的年份, 与文献的年份未必一致. 2.5 节用到的求最小平均成本的回路的算法引自 Karp [K78]. 另外, 解最小成本流问题时, 在实用的意义上, 即使现在, 用得最广的还是基于线性规划的算法. 这些算法很快. 因有现成程序 (比如 [IF91]), 很容易利用.

第 3 章 依据多面体方法的组合优化

作为求组合优化问题精确解的方法，基于多面体方法的分枝切割法很引人注目．该方法基于由可行域定义的整数多面体，解由其边界面构成的线性规划问题．这种方法在构造线性规划问题时结合了分枝定界法．本章以取得巨大成功的旅行商问题为具体例子，介绍其全貌．

3.1 多面体方法

组合优化问题包括，有关图论和网络的各种问题、求出机械作业的最优顺序的排序问题，决定设施最优地点的选址问题等为数众多的有重大实用价值的问题．但是，在计算复杂性理论 (参照本讲座第三卷「离散结构」第四卷「计算的理论」等) 意义下，其中大多数是极其困难的问题 (具体说来是 NP 困难的)，对所有问题的实例恒在多项式时间内解出被认为是不可能的．

然而，如果不是规模特别大的问题实例，或者虽然规模大但不是特别别扭的问题，实用上足可应付的情况也不少．为了达到这个目的，人们在尝试以下若干方法：

(a) 把给定的问题模型化为整数规划问题，利用针对一般整数规划的算法求解；

(b) 有效地利用对象问题的特殊性，利用针对该问题特殊

化后的整数规划算法求解；

(c) 不化为整数规划问题而直接通过分枝定界法解求解

在这些方法之中，关于最近很引人注目的**多面体方法** (polyhedral approach) 不断有重大成果出现. 这个方法是，把有待求解的问题模型化为整数规划问题之后，注意其整数解的凸包形成的整数多面体，解由其边界面所定义的线性规划问题. 这章在本节说明一般的思想之后，下一节开始对旅行商问题作详细介绍.

首先，假定对象问题已模型化为下面的**整数规划问题** (integer programming problem)[1)]

$$\text{(IP)}\quad \begin{aligned} &\text{目标函数:} && c^T x \quad \rightarrow \text{最小} \\ &\text{约束条件:} && Ax \leq b,\ x \geq 0 \\ & && x_j\ :\ \text{整数}\ (j=1,2,\ldots,N) \end{aligned} \tag{3.1}$$

这里，c 是 N 维成本向量，A 以及 b 是 $M\times N$ 以及 $M\times 1$ 的系数矩阵，右边的 0 是 0 元素组成的 N 维向量，x 是 N 维的向量变量.

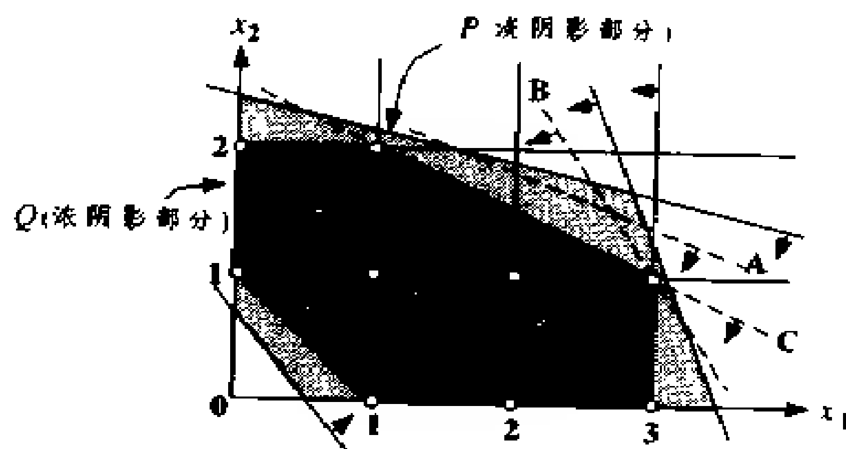


图 3.1 凸多面体 P 和整数多面体 Q (○ 是整数可行解)

1) 因所有变量都是整数变量，严格说来是全整数规划问题. 另外，也有仅仅一部分变量为整数变量的混合整数规划问题.

如果在问题 (IP) 中除去变量为整数的条件，就得到第 1 章讨论的线性规划问题．把它记为 (LP)．(LP) 的可行域是 N 维实数空间的**凸多面体** (polyhedron)[1]，把它记为 P．图 3.1 给出 $N=2$ 的情况下的例子，三条实线 (其可行侧用箭头表示) 连同条件 $x_j \geq 0$ 的五个一次不等式的共同部分给出了 P．问题 (IP) 的可行解是 P 内的整数点 (图 3.1 的小白圈)．这些整数点的凸包称为 (IP) 的**整数多面体** (integer polyhedron)，记为 Q．图 3.1 浓阴影的部分与此对应．Q 也是凸多面体．

给定凸多面体 $Q \subseteq R^N$ 的时候，一次不等式 $l^T x \leq l_0$ (这里，$l \in R^N, l_0 \in R$) 如果满足

$$Q \subseteq \{x \in R^N \mid l^T x \leq l_0\}$$

则称之为**有效**(valid) 的．对有效的一次不等式 $l^T x \leq l_0$，令

$$F = \{x \in R^N \mid l^T x = l_0\} \cap Q \tag{3.2}$$

(注意 F 的条件是等式)，F (或者对应的不等式 $l^T x \leq l_0$) 称为 Q 的**面** (face)．如果 F 进一步满足

$$F \neq \emptyset \text{ 以及 } F \neq W \tag{3.3}$$

则称为**真面** (proper face)．对真面 F，如果不存在满足 $F' \supsetneq F$ 的真面 F' (也就是，如果 F 是极大的)，则称 F (或者给出它的不等式) 为 Q 的**边界面** (facet)．图 3.1 里用虚线表示的不等式中，A 是有效的但不是真面．B 是真面但不是边界面．C 是边界面．

---

1) 参照 1.2 节第 8 页的脚注 1)．

一般把不等式 $l^T x \le l_0$ 仅用其系数向量表示为 (l, l_0). 把所有给出 Q 的边界面的不等式 (l, l_0) 的集合记为 $\mathcal{L}_Q$, 其阶数 $|\mathcal{L}_Q|$ 是有限的.

把话题转回到式 (3.1) 的问题 (IP). 如果可以求出对应于该 (IP) 的 $\mathcal{L}_Q$, 则可把 (IP) 写为

$$
\begin{array}{ll}
\text{目标函数:} & c^T x \longrightarrow \text{最小} \\
\text{约束条件:} & l^T x \le l_0, \quad ((l, l_0) \in \mathcal{L}_Q)
\end{array}
\tag{3.4}
$$

这个问题里没有整数条件, 是单纯的线性规划问题. 把式 (3.1) 的整数规划问题 (IP) 转化为式 (3.4) 的线性规划问题来解的方法称为多面体方法.

为了使多面体方法成功, 必须解决下述两个问题.

- 因整数多面体 Q 是由原来的凸多面体 P 间接定义的, 故生成 $\mathcal{L}_Q$ (或者它的一部分) 的算法是必要的.
- 即使生成了 $\mathcal{L}_Q$, 在很多情况下 $|\mathcal{L}_Q|$ 很大. 因此, 把它们全作为约束条件来解问题 (3.4) 的做法缺乏实用性.

本章以下利用旅行商问题的例子稍微详细看一看如何克服这些问题.

3.2 旅行商问题的多面体方法

3.2.1 旅行商问题及其模型化

作为困难组合优化问题的代表而知名的**旅行商问题** (traveling salesman problem), 是求出经过给定的 n 个点的最短**巡回路** (tour) 的问题. 记 $K_n = (V, E)$ 为具有 n 个节点的**完全无向图** (也即, $|V| = n$ 而且 E 由连接所有节点对的 $m = \binom{n}{2}$ 条

边构成)，各边 $e \in E$ 的长度为 $c(e)$ 的时候，在仅一次通过所有节点而回到出发点的回路 (称它为巡回路或者 **哈密尔顿回路** (Hamiltonian cycle)) 中，计算出最短者.

另外，这里的讨论以无向图为前提，对边 (v,w) 和边 (w,v) 不加区别，特别是 $c(v,w)=c(w,v)$ (以有向图为前提的旅行商问题也经常被讨论，但是本章不予处理)，对节点的子集合 $W \subseteq V$，采用记号

$$E(W)=\{(v,w)\in E \mid v,w\in W\} \tag{3.5}$$

另外，用 $x\in R^E$ 表示一个 m 维实数向量，其分量 $x(e)$ 对应于 E 的边 e. 进一步，对边的子集合 $F\subseteq E$，记

$$x(F)=\sum_{e\in F}x(e) \tag{3.6}$$

给定巡回路 τ 的时候，定义 τ 的特征向量 $x^\tau\in R^E$

$$x^\tau(e)=\begin{cases}1, & e\in\tau \text{ 的情况下}\\ 0, & e\notin\tau \text{ 的情况下}\end{cases}$$

K_n 的巡回路是从点 $v_1\in V$ 开始，以任意的顺序通过其它的 $n-1$ 个节点 $v_2,v_3,\cdots,v_n$ 而得到，所以合起来存在 $(n-1)!$ 个. 所有这些向量的凸包

$$Q^n=\mathrm{conv}\{x^\tau\in R^E \mid \tau\text{是}K_n\text{的巡回路}\} \tag{3.7}$$

就是旅行商问题的整数多面体.

各循环路 τ 里，一个节点连接着两条边，所以知道 Q^n 包含在多面体

$$Q_A^n=\{x\in R^E \mid Ax=2,\ 0\le x\le 1\} \tag{3.8}$$

里. 这里，$n\times m$ 矩阵 A 是 K_n 的 **邻接矩阵** (incidence matrix)，A 的各行对应节点 $v\in V$，各列对应边 $e\in E$，v 是 e 的端点时 (v,e) 的元素是 1，否则是 0. 另外，式 (3.8) 里的 0, 1, 2 分别是以标量 0, 1, 2 为所有元素的 (相应维数的) 向量. 因为一般很容易判定它们是向量还是标量，故在记号上不特地加以区别.

为简便起见把整数多面体 Q^n 的所有边界面的集合 $\mathcal{L}_{Q^n}$ 记为 $\mathcal{L}_n$ 即，

$$
\begin{aligned}
Q^n &= \{x \in R^E \mid l^T x < l_0,\ (l, l_0) \in \mathcal{L}_n\} \\
&= \{x \in Q^n_A \mid l^T x < l_0,\ (l, l_0) \in \mathcal{L}_n\}
\end{aligned}
$$

对应前节的式 (3.4)，利用 Q^n 可以把旅行商问题模型化为如下的线性规划问题.

$$
\begin{aligned}
&\text{目标函数:} \quad c^T x \longrightarrow \text{最小} \\
&\text{约束条件:} \quad Ax = 2, \\
&\qquad\qquad\quad\ l^T x \le l_0 \quad ((l, l_0) \in \mathcal{L}_n) \\
&\qquad\qquad\quad\ 0 \le x < 1
\end{aligned}
\tag{3.9}
$$

另一方面，具有整数多面体 Q^n 的整数规划问题 (前节的式 (3.1)) 是什么样的问题呢？首先，可以考虑

$$
\begin{aligned}
&\text{目标函数:} \quad c^T x \longrightarrow \text{最小} \\
&\text{约束条件:} \quad Ax = 2 \\
&\qquad\qquad\quad\ x(e) = 0, 1 \quad (e \in E)
\end{aligned}
\tag{3.10}
$$

由于在它的可行解中包含有图 3.2 那样由多个子巡回路组成的巡回路，所以需要能排除这种巡回路的约束条件.

图 3.2 由子巡回路组成的问题 (3.10) 的可行解

图 3.3 问题 (3.12) 中除去整数条件情况下的可行解

考虑一个子巡回路，经由它的(非空)节点集合设为 $W \subsetneq V$. 为禁止这个子巡回路，只要加上条件

$$x(E(W)) \le |W| - 1 \tag{3.11}$$

即可. 这里，根据本节开始叙述的记法，$x(E(W))$ 表示 $\sum_{e\in E(W)} x(e)$. 因为经由全节点的巡回路中，连接子集合 W 内的节点对之边最多只有 $|W|-1$ 条，所以这个条件仅排除子巡回路而不排除正确的巡回路. 称式 (3.11) 为**排除子巡回路约束** (subtour elimination constraint).

式 (3.10) 里加上式 (3.11) 后的整数规划问题

$$\begin{aligned} &\text{目标函数:} && c^T x \longrightarrow \text{最小} \\ &\text{约束条件:} && Ax = 2 \\ & && x(E(W)) \le |W| - 1 \quad (W \subsetneq V, W \neq \emptyset) \\ & && x(e) = 0, 1 \quad (e \in E) \end{aligned} \tag{3.12}$$

正确给出旅行商问题的模型. 也就是，这个问题的的整数多面体是 Q^n. 变量的整数条件是本质的，如果除去它，则可能出现图 3.3 那样的解，因而未必给出巡回路.

3.2.2 适当地生成边界面

Q^n 的边界面集合 $\mathcal{L}_n$ 里究竟包含些什么东西呢？像后面 3.3 节所讲的那样，前面排除子巡回路的条件 (3.11) 是其中一部分，除此之外还有很多边界面. 到现在为止还没有完全弄清 $\mathcal{L}_n$，而仅仅知道它的一部分 $\mathcal{L}'_n$. 故，以 $\mathcal{L}'_n$ 的边界面为约束条件来解线性规划问题. 但是，即使限定在 $\mathcal{L}'_n$ 里，边界面的个数也很庞大，全以显式处理是不现实的. 因此，下面适当逐步追加 $\mathcal{L}'_n$ 的一部分条件来求解，这个过程是有效的.

过程 FACET (生成边界面以及解 LP)

第一步 (初始化): $\mathcal{L} \leftarrow \emptyset$

第二步 (解 LP): 基于 $\mathcal{L}$ 解线性规划问题

$$
\begin{aligned}
&\text{目标函数:} \quad c^T x \rightarrow \text{最小} \\
&\text{约束条件:} \quad Ax = 2,\ 0 \le x \le 1 \\
&\qquad\qquad\quad\ l^T x \le l_0 \quad ((l, l_0) \in \mathcal{L})
\end{aligned}
\tag{3.13}
$$

求出它的最优解 x

第三步 (生成边界面): $\mathcal{L}'_n$ 的边界面中如果有 x 所不能满足者，则求出其中几个. 如果没有这样的边界面则结束计算. 在找到边界面的情况下，把它追加到 $\mathcal{L}$ 中，再回到第二步.

第三步中求边界面的问题称为**生成边界面的问题** (facet identification problem) 或者**边界面分离问题** (facet separation problem). 其具体求法将在 3.4 节加以说明.

这种在线性规划问题里追加有效的一次不等式而得到整数解的方法一般称为**割平面法** (cutting plane method). 上面的过程 FACET 有一个特征，它仅仅使用表现为一次不等式的边界面. 因为边界面给出的是整数多面体的极大面，故不存在比它更强的有效条件. 可以说正因为此它有很强的效果.

3.2.3 分枝切割法

因为 $\mathcal{L}'_n$ 是有限集合，所以过程 FACET 在有限次迭代结束. 但是，当 n 很大时，$|\mathcal{L}'_n|$ 相当大，而且即使执行到最后都未必能保证得到最短巡回路 (因为 $\mathcal{L}'_n \subsetneq \mathcal{L}_n$)，所以可以采

取在适当的时候中止迭代的手段. 因此有必要把**分枝定界法** (branch-and-bound method) 引入过程 FACET.

分枝定界法是在所有的整数解中有系统地生成可能成为最优整数解者并加以检验的过程. 这是可以适用于各种组合优化问题的有一般性的想法. 但是, 在这里我们仅限于对旅行商问题, 并仅对引入过程 FACET 后的**分枝切割法** (branch-and-cut method) 加以说明.

把在计算的某个时刻边界面的集合记为 $\mathcal{L}$, 由 $x(e) = 0$ ($x(e) = 1$) 所固定的边 e 的集合记为 F_0 (F_1) 相应地可以定义线性规划问题

$$
\begin{aligned}
P(\mathcal{L}, F_0, F_1): \quad & \text{目标函数}: & & c^T x \longrightarrow \text{最小} \\
& \text{约束条件}: & & Ax = 2,\ \ 0 \le x \le 1 \\
& & & l^T x \le l_0 \ \ ((l, l_0) \in \mathcal{L}) \qquad (3.14) \\
& & & x(e) = 0 \ \ (e \in F_0) \\
& & & x(e) = 1 \ \ (e \in F_1)
\end{aligned}
$$

设 $P(\mathcal{L}, F_0, F_1)$ 的实数最优解为 $\bar{x}$. 另外, 由适当的近似解法或者目前的分枝定界法的计算所得到的巡回路中最短者 x^* 叫作**暂定解** (incumbent), 其值 $z^* = c^T x^*$ 叫作**暂定值** (incumbent value). 此时, 有如下两个性质:

1. 如果 $\bar{x}$ 给出巡回路, 则它是 F_0 和 F_1 的条件下的最短巡回路;

2. 在 $P(\mathcal{L}, F_0, F_1)$ 里不存在有可行解的情况下, 或者在成立 $c^T \bar{x} \ge z^*$ 的情况下, $P(\mathcal{L}, F_0, F_1)$ 不可能给出比 x^* 更短的巡回路.

不管在哪种情况下, 即使查询满足 $F_0' \supseteq F_0$ 和 $F_1' \supseteq F_1$ 的

$P(\mathcal{L}, F_0', F_1')$ 也不可能得出比 x^* 或 x 更短的巡回路，因而可以排除出以后的考虑范围.

分枝切割法是生成满足 $F_0 \cap F_1 = \emptyset$ 的一对 $< F_0, F_1 >$ 后，再求解 $P(\mathcal{L}, F_0, F_1)$ 的过程. 当然，如果查询所有这样的 $< F_0, F_1 >$，可以求出最短巡回路 (这仅仅是列举法). 实际上根据上面的考察，可以通过仅仅查询对 $< F_0, F_1 >$ 的很小一部分而达到目的. 为生成 $< F_0, F_1 >$，从 $F_0 = \emptyset$，$F_1 = \emptyset$ 开始，在得不到上面的结论 1 或者 2 的情况下，选取适当的 $e \notin F_0 \cup F_1$，替换成由

$$\begin{aligned} &F_0' := F_0, \quad F_1' := F_1 \cup \{e\} \\ &F_0'' := F_0 \cup \{e\}, \quad F_1'' := F_1 \end{aligned} \tag{3.15}$$

确定的 $< F_0', F_1' >$ 和 $< F_0'', F_1'' >$ 不断反复进行这种操作. 称这个 e (或者由此而确定的 $x(e)$) 为**分枝变量** (branching variable)

$< F_0, F_1 >$ 的生成状况可以用图 3.4 的生成树表示. 当然，根据上面的检验 1 和 2，实际上有很多 $< F_0, F_1 >$ 不会被生成，由此提高计算效率.

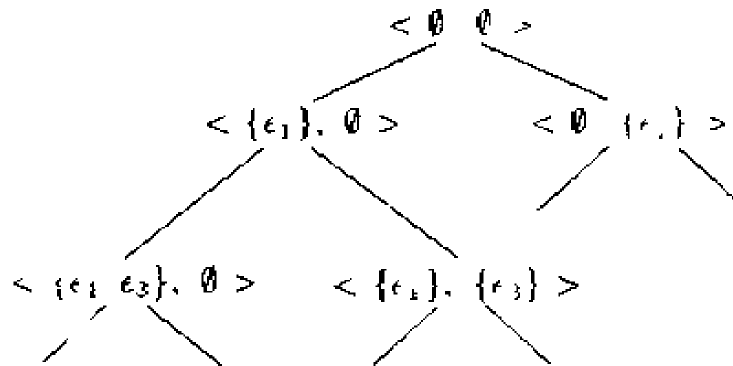


图 3.4 产生子问题 $< F_0, F_1 >$ 的生成树

上面的过程可以概括如下. 其中，S 是存放在此之后应该考虑的 $< F_0, F_1 >$ 的清单.

算法 BCUT (分枝切割法)

第一步 (初始化): 令 $\mathcal{S} = \{< \emptyset, \emptyset >\}$, $\mathcal{L} := \emptyset$. 利用适当的近似解法求出巡回路，记忆暂定解 x^*, 同时记忆其值 $c^T x^*$ 作为暂定值 z^*

第二步 (搜索): 如果 $\mathcal{S} = \emptyset$, 则结束计算，x^* 是最短巡回路. 如果 $\mathcal{S} \neq \emptyset$, 则选取适当的对 $< F_0, F_1 > \in \mathcal{S}$, 令 $\mathcal{S} := \mathcal{S} - \{< F_0, F_1 >\}$.

第三步 (LP 解): 解线性规划问题 $P(\mathcal{L}, F_0, F_1)$. 如该问题不存在可行解则返回到第二步. 否则，求出其最优解，记为 $\bar{x}$

第四步 (下界值检验): 如果 $c^T x \geq z^*$ 则回到第二步.

第五步 (生成边界面): 求出几个 $\bar{x}$ 不满足的 $(l, l_0) \in \mathcal{L}'_n$.

第六步 ($\mathcal{L}$ 的收敛判定): 如果不存在这种 $(l, l_0) \in \mathcal{L}'_n$ 则进入第七步. 如果存在，则把它们加进 $\mathcal{L}$ 后回到第三步.

第七步 (x^* 的更新): 如果 $\bar{x}$ 是巡回路，则令 $x^* = \bar{x}$, $z^* := c^T \bar{x}$ 后回到第二步.

第八步 (分枝操作): 选取一个满足 $0 < \bar{x}(e) < 1$ 的 $e \in E$, 在 $\mathcal{S}$ 里加进 $< F_0, F_1 \cup \{e\} >$, $< F_0 \cup \{e\}, F_1 >$ 回到第二步.

为使算法 BCUT 作为具体的程序得以实现，必须确定第一步中 x^* 的计算法，第二步中 $< F_0, F_1 > \in \mathcal{S}$ 的选定法，第五步中边界面的生成法，第八步中 $e \in E$ 的决定法. 不同的方法导致计算效率的巨大差异.

关于第一步中近似解 x^* 的求法，也就是旅行商问题的近似解法，已作了广泛研究. Lin 和 Kernighan 的算法 [LK 73] 等广为人知，本书省略其详细说明. 根据这些方法，通常可以有效地求出与最优值的相对误差在几个百分点以内的巡回路.

关于第五步中生成边界面的方法，在 3.3 节和 3.4 节作详细说明．另外，这样生成的边界面在所有子问题 $P(\mathcal{L}, F_0, F_1)$ 里可以共同利用，所以随着计算过程的进展，数据处理也变得越简单，这也成为分枝切割法的一个特征．

其次，注意线性规划问题 $P(\mathcal{L}, F_0, F_1)$ 在很多情况下具有特大规模．变量 $x(e)$ 的个数有 $m = n(n-1)/2$, 比如 $n = 1000$ 的情况下，大约成为 500,000 个．另外，由于边界面的追加，约束条件的个数也随着第六步的迭代急速增大．多次求解这种大规模问题时，把变量和约束条件限制在真正与计算相关者来求解的技巧是很有效的．其内容在 3.5 节叙述．

关于第二步和第八步的考察在 3.6 节进行．

3.3 旅行商问题的边界面

作为多面体方法的核心，关于边界面长期以来有很详细的研究．像本节后半部分要讲的那样，式 (3.11) 的排除子巡回路的约束便是边界面的一例．除此之外，还有以下各种边界面．

3.3.1 梳子约束 (comb inequality)

在被称为**把手** (handle) 的集合 $H \subseteq V$ 和被称为**齿** (tooth) 的集合 $T_i \subseteq V$ $(i = 1, 2, \ldots, k)$ 满足条件

$$\begin{aligned}
&\text{(a)} \quad T_i \cap T_j = \emptyset \quad (i \neq j) \\
&\text{(b)} \quad |H \cap T_i| \geq 1 \quad (i = 1, 2, \ldots, k) \\
&\text{(c)} \quad |T_i \setminus H| \geq 1 \quad (i = 1, 2, \ldots, k)^{1)} \\
&\text{(d)} \quad k: \text{奇数}
\end{aligned} \tag{3.16}$$

---

1) $T_i \setminus H$ 表示包含在 T_i 内但不属于 H 的元素集合．

的时候 (参照图 3.5)，由

$$x(E(H)) + \sum_{i=1}^{k} x(E(T_i)) \le |H| + \sum_{i=1}^{k}(|T_i| - 1) - \frac{k+1}{2} \tag{3.17}$$

给出这个约束．特别是在 $k = 1$、$|H| = 1$ 的情况下，这个条件成为

$$x(E(T_1)) \le |T_1| - 1$$

它即为排除子巡回路的约束 (3.11)．另外，在所有的齿 T_i 满足

$$|T_i| = 2 \quad (i = 1, 2, \ldots, k) \tag{3.18}$$

的时候，特别称它为 **2- 匹配约束** (2-matching inequality)．

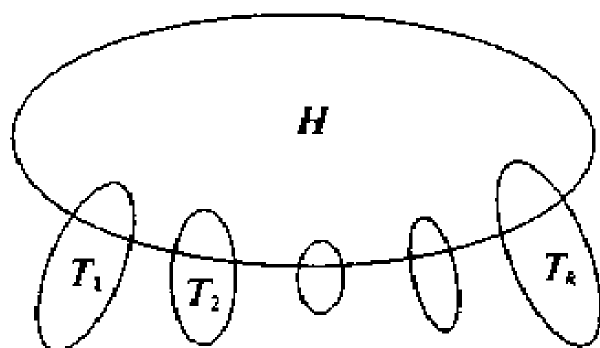


图 3.5 梳子约束

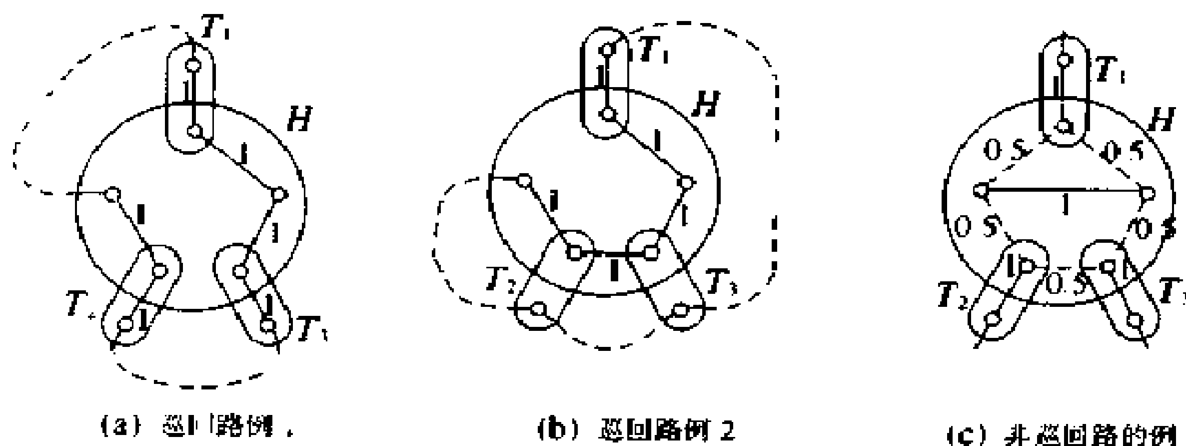


图 3.6 梳子约束和巡回路

通过图 3.6 来考虑一下梳子约束 (3.17) 的意思. 图 3.6 的 (a) 和 (b) 是巡回路 (对应于实线和虚线的边 e, $x(e)=1$ 其它的 $x(e)=0$), (c) 不是巡回路. 考虑到 $|H|=5$, $T_i=2$ $(i=1,2,3)$, 不论是在 (a) (b) (c) 的哪一种情况下梳子约束右边的值都是 6. 另一方面, 梳子约束的左边, 在 (a) 的情况下, $x(E(H))=3$, $x(E(T_i))=1$ $(i=1,2,3)$, 因而取值 6. 在 (b) 的情况下, $x(E(H))=4$, $x(E(T_1))=1$, $x(E(T_2))=x(E(T_3))=0$ 因而取值 5. 也即满足梳子约束条件. 但是在 (c) 的情况下, $x(E(H))=3.5$, $x(E(T_i))=1$ $(i=1,2,3)$, 左边的值是 6.5, 也即这个解不满足梳子约束. 梳子约束具有排除像 (c) 那样的非巡回路的效果.

梳子约束是有效的条件 (其定义在 3.1 节, 也就是任意的巡回路满足梳子约束). 这个事实可以如下证明. 这里, 用 $E(v,V-v)$ 表示以 v 为一个节点的边的集合 (组成以 v 为中心的星). 设 x 给出巡回路的解, 则成立

$$
\begin{aligned}
&2\{x(E(H))+\sum_{i=1}^{k}x(E(T_i))\}\\
\le&\sum_{v\in H}x(E(v,V-v))\\
&+\sum_{i=1}^{k}\{x(E(T_i))+x(E(T_i-H))+x(E(T_i\cap H))\}\\
\le&2|H|+\sum_{i=1}^{k}\{|T_i|-1+|T_i-H|-1+|T_i\cap H|-1\}\\
\le&2\{|H|+\sum_{i=1}^{k}(|T_i|-1)\}-k
\end{aligned}
\tag{3.19}
$$

第一个不等式可以如下理解. $\sum_{v\in H}x(E(v,V-v))$ 对 $E(H)$ 内的每条边计数两次, 对从 H 出来的每条边计数一次. 比较边

集合 $E(T_i - H) \cup E(T_i \cap H)$ 和 $E(T_i)$. 包含于后者的边, 如果不在前者里面, 则一定是 $E(T_i \cap H, T_i - H)$ 的边, 它已经作为 $\sum_{v \in H} x(E(v, V - v))$ 的一部分计数了. 第二个不等式是因为约束 $Ax = 2$, 以及对于任意的 $W \subsetneq V$, 巡回路最多仅仅使用了 $E(W)$ 的 $|W| - 1$ 条边 (也即排除子巡回路的约束). 第三个不等式是整理前式的结果. 全体除 2, 再对两边取整数部分就得到梳子约束 (3.17). 梳子约束的最后加上条件 k (奇数) 是因为, k 是偶数的时候, 即使对两边取整数部分后, 约束条件也得不到加强, 并不一定给出边界面.

梳子约束是真面 (其定义见 3.1 节), 这个事实可以如下理解. 用图 3.6 的例来说, 因为存在像 (a) 那样以等号形式满足梳子约束的巡回路, 还存在像 (b) 那样以严格不等号形式满足的巡回路. 进一步可以证明梳子约束是边界面. 因其证明有点复杂, 在此省略. 可以采取如 3.3.3 里叙述的, 和针对排除子巡回路约束的边界面性的证明一样的思路.

3.3.2 团树约束 (clique tree inequality)

作为对梳子约束的推广, 考虑把手的集合 $H_j \subseteq V$ $(j = 1, 2, \ldots, h)$ 和齿的集合 $T_i \subseteq V$ $(i = 1, 2, \ldots, k)$. 当它们满足以下条件时, 称为**团树**

(a) $H_j \cap H_{j'} = \emptyset \quad (j \neq j')$

(b) $T_i \cap T_{i'} = \emptyset \quad (i \neq i')$

(c) $2 < |T_i| \leq n - 2 \quad (i = 1, 2, \ldots, k)$

(d) $|T_i - \cup_{j=1}^{h} H_j| \geq 1 \quad (i = 1, 2, \ldots, k)$

(e) $|I_j| \geq 3$ 而且是奇数 $(j = 1, 2, \ldots, h)$, 这里, $I_j = \{i | H_j \cap T_i \neq \emptyset\}$

(f) 如果 $H_j \cap T_i \neq \emptyset$ 则 $H_j \cap T_i$ 是关节集合.

在如上给出节点子集合 H_j ($j=1,2,\ldots,h$) 和 T_i ($i=1,2,\ldots,k$) 的时候，考虑具有节点集合 $V^*=(\cup_{j=1}^{h}H_j)\cup(\cup_{i=1}^{k}T_i)$ 和边集合 $E^*=(\cup_{j=1}^{h}E(H_j))\cup(\cup_{i=1}^{k}E(T_i))$ 的图 $G^*=(V^*,E^*)$．不失一般性可以假设 G^* 连通．否则，只要把以下的讨论用在每个连通部分即可．某个节点的集合 $W\subseteq A$ 是所谓的**关节集合** (articulation set)，是指 G^* 的由 V^*-W 生成的子图不连通．图 3.7 示意了团树的一个例子．图中阴影集合是 T_i，白领域表示 H_j

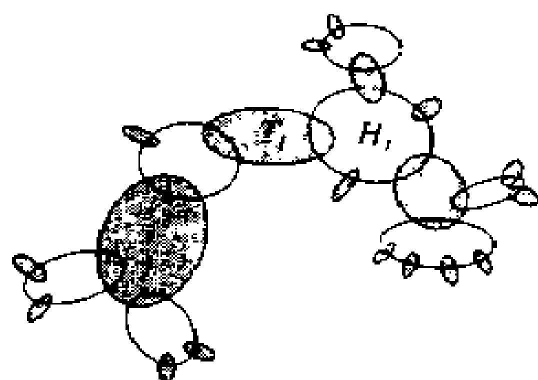


图 3.7 团树

团树约束可以写成下面的形式．

$$\sum_{j=1}^{h}x(E(H_j))+\sum_{i=1}^{k}x(E(T_i))\le\sum_{j=1}^{h}|H_j|+\sum_{i=1}^{k}(|T_i|-t_i)-\frac{k+1}{2}\quad(3.20)$$

这里，

$$t_i=|\{j|H_j\cap T_i\ne\emptyset,\ j=1,2,\ldots,h\}|\quad(i=1,2,\ldots,k)\quad(3.21)$$

当 $h=1$ 时，团树约束相当于梳子约束 (3.17)．由此可以知道团树约束是梳子约束的推广．团树约束的有效性也可以同样证明．进一步，其边界面性也得到了证明．

3.3.3 边界面性的证明

为证明某个有效约束是整数多面体的边界面，一般有必要返回到有关的定义，并进行几何学的讨论．这里，作为这种证明的例子，证明排除子巡回路约束 (3.11) 的边界面性．

为记法简单起见，把排除子巡回路约束之

$$x(E(W)) \le |W| - 1$$

写成下式

$$b^T x \le b_0 \tag{3.22}$$

这里，b 是 $m(= n(n-1)/2)$ 维向量，b_0 是正整数．为证明式 (3.22) 是边界面，像 3.1 节所述的那样，只要证明集合

$$F = \{x \in R^m | b^T x = b_0\} \cap Q^n$$

的极大性．也即，对给出真面的任意的有效约束 $d^T x \le d_0$，面

$$F' = \{x \in R^m | d^T x = d_0\} \cap Q^n$$

具有如下性质．

$$F' \supseteq F \Rightarrow F' = F \tag{3.23}$$

首先，证明性质 (3.23) 的一个充分条件：在 $F' \supseteq F$ 的时候，对某个 $\alpha > 0$ 和 $\lambda \in R^n$，成立

$$d^T = \alpha b^T + \lambda^T A \tag{3.24}$$

这里，A 是 (3.8) 的邻接矩阵．注意这个时候如果 $F' \supseteq F$，则

$$d_0 = \alpha b_0 + 2\sum_{i=1}^{n} \lambda_i$$

这是因为，对于 $x \in F$，一定成立 $x \in F'$，也即

$$d^T x = \alpha b^T x + \lambda^T A x = \alpha b_0 + 2\sum_{i=1}^{n} \lambda_i = d_0$$

这里利用了式(3.8)里的条件 $Ax = 2$. 如果 d 和 d_0 成立上述关系，则

$$d^T x = d_0 \Rightarrow \alpha b^T x + \lambda^T A x = \alpha b_0 + 2\sum_{i=1}^{n} \lambda_i$$
$$\Rightarrow \alpha b^T x = \alpha b_0 \Rightarrow b^T x = b_0$$

这意味着，如果 $x \in F'$ 则 $x \in F$（也即 $F' \subseteq F$)，证明了性质(3.23)成立

因此，以下证明在 $F' \supseteq F$ 的时候成立式(3.24). 这里，为简单起见，以下仅仅用下标 i 表示 v_i. 另外，选取

$$W = \{1, 2, \ldots, k\} \tag{3.25}$$

作为给出式(3.22)的集合 W，进一步假定 $n \geq 6$ 而且 $2 < k \leq n - 3$. 考虑 n 条边形成的集合

$$E_0 = \{(1, i) | i = 2, 3, \ldots, n\} \cup \{(2, 3)\}$$

E_0 构成连结 n 个节点的生成树和一个奇数长的回路 (1, 2), (2, 3), (3, 1)，所以邻接矩阵 A 中仅选取与 E_0 对应的列得到的 A_{E_0} 是非奇异的(这是离散数学里有名的性质，其证明也不难). 于是，适当选取 $\overline{\lambda} \in R^n$，可以使

$$\overline{d}^T = d^T + \overline{\lambda}^T A$$

满足条件

$$\overline{d}_{1i} = b_{1i} \qquad (i = 2, 3, \ldots, n)$$
$$\overline{d}_{23} = b_{23}$$

这样确定的 d^T 如果满足条件 (3.24), 则可以断言原来的 d^T 也满足条件 (3.24). 所以，为简单起见，一开始就假定

$$\begin{aligned} d_{1i} &= b_{1i} \quad (i = 2, 3, \ldots, n) \\ d_{23} &= b_{23} \end{aligned} \tag{3.26}$$

这里，根据 W 的定义 (3.25) 和式 (3.22),

$$b_{1i} = \begin{cases} 1, & i = 2, 3, \ldots, k \text{ 的情况下} \\ 0, & i = k+1, k+2, \ldots, n \text{ 的情况下} \end{cases}$$

$$b_{23} = 1$$

其次，考虑巡回路 $\tau_1 = (1, 3, 4, \ldots, j, 2, j+1, j+2, \ldots, n)$ 和 $\tau_2 = (1, j, j-1, \ldots, 3, 2, j+1, j+2, \ldots, n)$. 这里，$j$ 是满足 $2 < j < k$ 的任意的节点．因为它们使用了 $E(W)$ 的 $k-1$ 条边，所以它们的特征向量 x^{τ_1} 和 x^{τ_2} 以等号的形式满足 $b^T x \leq b_0$. 根据 $F' \supseteq F$, 也以等号形式满足 $d^T x \leq d_0$. 由

$$0 = d^T x^{\tau_1} - d^T x^{\tau_2} = d_{13} + d_{2j} - d_{1j} - d_{23} = d_{2j} - 1 \quad (\text{根据式 (3.26)})$$

得到 $d_{2j} = 1$ $(2 < j \leq k)$. 一般，利用 $\tau_3 = (1, 2, \ldots, i-1, i+1, \ldots, j, i, j+1, j+2, \ldots, n)$ 和 $\tau_4 = (1, j, j-1, \ldots, i+1, 2, 3, \ldots, i, j+1, j+2, \ldots, n)$ 对 $1 \leq i < j \leq k$ 进行同样的讨论，则可断言

$$d_{ij} = 1 \quad (1 \leq i < j \leq k) \tag{3.27}$$

另外，从巡回路 $\tau_5 = (1, 2, \ldots, n)$ 和 $\tau_6 = (2, 1, 3, 4, \ldots, n)$ 得出

$$0 = d_{23} + d_{1n} - d_{13} - d_{2n}$$

也即得出 $d_{2n} = 0$. 进一步，如果在 τ_5 和 τ_6 中把 2 和 n 起的作用换到一般的 i 和 j 进行同样的讨论，就可以证明

$$d_{ij} = 0 \quad (1 \leq i \leq k, k+1 \leq j \leq n) \tag{3.28}$$

最后、从巡回路 $\tau_7 = (1,2,k+2,k+1,n,n-1,\ldots,k+3,3,4,\ldots,k)$ 和 $\tau_8 = (1,2,k+1,n,n-1,\ldots,k+2,3,4,\ldots,k)$ 得出

$$0 = d_{k+1k+2} - d_{k+2k+3}$$

根据同样的讨论，知道存在有满足

$$d_{ij} = \beta \quad (k+1 \leq i < j \leq n) \tag{3.29}$$

的常数 β

合并式 (3.27), (3.28) 和 (3.29) 就有

$$\begin{aligned} d^T x &= x(E(W)) + \beta x(E(V-W)) \\ d_0 &= |W| - 1 + \beta(|V-W| - 1) \end{aligned} \tag{3.30}$$

d_0 的值可以通过、比如 τ_5 以等号形式满足 $d^T x \leq d_0$ 而导出. 为求出 β 的值的范围，进一步考察巡回路 $\tau_9 = (1,n,2,3,\ldots,k,k+1,\ldots,n-1)$, 成立有

$$d^T x^{\tau_9} = |W| - 2 + \beta(|V-W| - 2) \leq |W| - 1 + \beta(|V-W| - 1)\ (= d_0)$$

故 $\beta \geq -1$. 其结果，令

$$\alpha = 1 + \beta \quad (\geq 0)$$

$$\lambda_i = \begin{cases} -\beta/2, & i = 1,2,\ldots,k \text{ 的情况下} \\ \beta/2, & i = k+1,k+2,\ldots,n \text{ 的情况下} \end{cases}$$

式 (3.30) 的 d^T 可以写成

$$d^T = \alpha b^T + \lambda^T A$$

也即证明了式 (3.24). 另外，$\beta = -1$ 也即 $\alpha = 0$ 的情况下，$d^T = \lambda^T A$ 与 F' 是 Q^n 的真面相矛盾 (因不存在 $x \notin F'$ 的 $x \in Q^n$), 故可以不必考虑.

概括上面的结果，证明了排除子巡回路约束 (3.22) 是 Q^n 的边界面.

3.4 生成边界面的方法

分枝切割法 BCUT 的第五步中，在 $P(\mathcal{L}, F_0, F_1)$ 的最优解 x 不是巡回路的情况下，要求生成几个排除 $\overline{x}$ 的边界面约束. 这是生成边界面的问题，需要有高效率的算法. 本节针对排除子巡回路约束以及 2 匹配约束介绍其算法.

3.4.1 排除子巡回路约束

首先构造基于 $\overline{x}$ 的图

$$G_{\overline{x}} = (V, E_{\overline{x}})$$

$$E_{\overline{x}} = \{e \in E \mid \overline{x}(e) > 0\}$$

把 $\overline{x}(e)$ 看成边 e 的容量，定义网络

$$N_{\overline{x}} = (G_{\overline{x}}, \overline{x}) \tag{3.31}$$

对某个节点集合 W (这里，$W \subsetneq V, W \neq \emptyset$)，连接 W 和 $V - W$ 的边的集合 (参照图 3.8)

$$E(W, V - W) = \{(v, w) \in E_{\overline{x}} | v \in W, w \in V - W\} \tag{3.32}$$

称为 $N_{\overline{x}}$ 的截 (cut)，称

$$\overline{x}(E(W, V - W)) = \sum_{e \in E(W, V - W)} \overline{x}(e) \tag{3.33}$$

为其容量

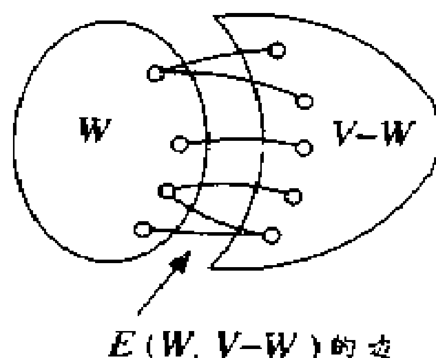


图 3.8 截 $E(W,V-W)$

这里，对各 $v\in W$，考虑在连接 v 的边集合 $E(v,V-v)$ 上 $x(e)$ 的和，可得如下二式

$$\begin{aligned}\sum_{v\in W} x(E(v,V-v)) &= 2x(E(W)) + x(E(W,V-W)) \\ \sum_{v\in W} x(E(v,V-v)) &= 2|W| \quad (\text{因为条件 } Ax=2)\end{aligned} \tag{3.34}$$

因此，作为排除子循环路约束 (3.11) 的

$$x(E(W)) \le |W|-1$$

与如下条件等价，

$$x(E(W,V-W)) \ge 2$$

于是，可以知道存在排除 x 的子巡回路约束的充分必要条件是，N_x 里存在容量比 2 小的截. 这可以通过求出 N_x 的最小容量截来判定.

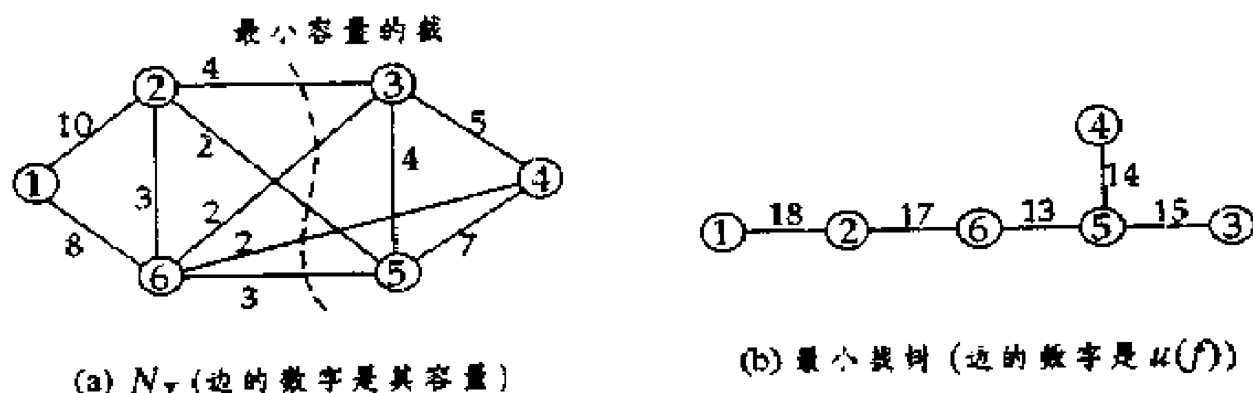
作为网络的基本问题，求网络的最小容量截的问题已有广泛研究. Gomory 和 Hu 提出了一个有代表性的算法. 它是用某种方法选取 $n-1$ 组节点对 (s,t) (其详细过程在此省略)，通

过解 s 到 t 的最大流问题 (参照 2.2.2), 来求出分离 s 和 t 的最小容量截 (这是基于网络理论中著名的**最大流最小截** (max flow min-cut) **定理**). 其结果, 求出了具有下述性质 (i), (ii) 的**最小截树** (minimum cut tree) T.

(i) $T=(V,F)$ 是生成树, 各边 $f \in F$ 具有容量 $u(f)$.

(ii) 对任意的 $v, w \in V$, 记 T 上从 v 到 w 的唯一路上边的最小容量为 $u_{v,w}$. $u_{v,w}$ 等于原网络 N_T 里分离 v 和 w 的截的最小容量.

图 3.9 给出了 N_T 以及于它对应的 T 的例. 特别是, 如果在整个 T 中求出最小容量的边 f, 则 $u(f)$ 等于 N_T 的最小截的容量. Gomory 和 Hu 的算法在强多项式时间 (对应最大流问题的时间量 $\times(n-1)$) 内完成.



(a) N_T (边的数字是其容量)

(b) 最小截树 (边的数字是 $u(f)$)

图 3.9 Gomory 和 Hu 的最小截树

3.4.2 2- 匹配约束

在节点集合 $H \subseteq V$ 和 k 条边 e_i $(i=1,2,\ldots,k)$ 满足条件

(a) e_i 和 e_j $(i \neq j)$ 没有共同节点,

(b) 各边 e_i 的一个端点在 H 内, 另一个在 H 外,

(c) k 奇数,

的时候，这个 2- 匹配约束可以写成

$$x(E(H)) + \sum_{i=1}^{k} x(e_i) \le |H| + \frac{k-1}{2} \tag{3.35}$$

(参照式 (3.17), (3.18)).

这里、利用松弛变量 $t(e)$ 把式 (3.8) 改写为

$$Ax = 2$$

$$x(e) + t(e) = 1, \quad x(e) \ge 0, \quad t(e) \ge 0 \quad (e \in E)$$

在条件 $Ax = 2$ 中，把与 H 内的节点对应的等式加起来，进一步对 $x(e) + t(e) = 1$ 关于 e_i $(i = 1, 2, \ldots, k)$ 取和，根据同式 (3.34) 一样的考察有

$$2x(E(H)) + x(E(H, V-H)) + \sum_{i=1}^{k} x(e_i) + \sum_{i=1}^{k} t(e_i) = 2|H| + k$$

也即得到，

$$2\left(x(E(H)) + \sum_{i=1}^{k} x(e_i)\right)$$

$$= 2|H| + k - x(E(H, V-H)) - \sum_{i=1}^{k} t(e_i) + \sum_{i=1}^{k} x(e_i)$$

利用这个关系式，可以把 2- 匹配条件 (3.35) 改写成

$$\sum_{i=1}^{k} (1 - x(e_i)) + x(E(H, V-H)) - \sum_{i=1}^{k} x(e_i) \ge 1 \tag{3.36}$$

进一步，对 e_i $(i = 1, 2, \ldots, k)$，根据前面的条件 (b)，可以表示为

$$\sum_{i=1}^{k} (1 - x(e_i)) + \sum_{e \in E(H, V-H) - \{e_1, e_2, \ldots, e_k\}} x(e) \ge 1$$

该式的左边是在截 $E(H, V-H)$ 的边 e 中，对 $e_i\ (i=1,2,\cdots,k)$ 取 $(1-\bar{x}(e_i))$，对其它的边取 $\bar{x}(e)$ 而求的和.

结果，把求排除 $\bar{x}$ 的 2- 匹配约束的问题解释成为在 $N_{\bar{x}}$ 的截 $E(H, V-H)$ 里，把其中奇数条边 e 的容量从 $\bar{x}(e)$ 变更为 $(1-\bar{x}(e))$ 的时候是否存在具有容量比 1 更小的截的问题.

为使问题进一步明了，在各边 $e=(v,w)\in E_{\bar{x}}$ 上添加新点 y_e、y'_e，分成三条边

$$e^{(1)}=(v,y_e),\quad e^{(2)}=(y_e,y'_e),\quad e^{(3)}=(y'_e,e)$$

把得到的图记为 $G'_{\bar{x}}=(V',E'_{\bar{x}})$ 进一步设定各边的容量为

$$\bar{x}(e^{(1)})=\bar{x}(e^{(3)})=x(e)$$

$$\bar{x}(e^{(2)})=1-\bar{x}(e)$$

把得到的网络表示为 $N'_{\bar{x}}=(G'_{\bar{x}},\bar{x})$. 这样一来，$v\in H$ 以及 $w\in V-H$ 的时候，$e=(v,w)$ 是否包含在奇数条边的集合 $\{e_1,e_2,\cdots,e_k\}$ 里的问题可以转换为三条边 $e^{(1)}$、$e^{(2)}$、$e^{(3)}$ 形成的部分是用 $e^{(2)}$ 切开 (即，$e\in\{e_1,e_2,\cdots,e_k\}$) 还是用 $e^{(1)}$ 切开 (即，$e\notin\{e_1,e_2,\cdots,e_k\}$) 的形式. 因此，把 G'_x 里原来 $G_{\bar{x}}$ 有的节点作为偶点，把新加入的点 y_e 和 y'_e 作为奇点而加以区分. 这样一来，待解的问题归结为在 $N'_{\bar{x}}$ 里，在 H' (同样地，$V'-H'$) 包含有奇数个奇点的条件下求出最小容量的截 $E(H',V'-H')$ 如果该容量比 1 小，则把 H' 中偶点的集合作为 H、$E(H',V'-H')$ 里包含的 $e^{(2)}$ 型的边 (y_e,y'_e) 所对应的 e 的集合作为 $\{e_1,e_2,\cdots,e_k\}$，得到了排除 $\bar{x}$ 的 2- 匹配约束. 如果最小容量在 1 以上，则 $\bar{x}$ 满足所有的 2- 匹配约束.

上面的问题带有包含奇数个奇点的条件，所以与通常的最小容量截问题不一样. 但是已经知道，经过下述步骤它可以在多项式时间内解出.

1. 仅仅针对 $N_x^{\prime}$ 的奇点利用上述 Gomory 和 Hu 的算法，构造出最小容量截的树 T' (T' 的各节点对应于 $N_x^{\prime}$ 的奇点).

2. 从 T' 中除去一条边 f 后把 T' 分成两个子树，在使得两个子树的节点树都是奇数的 f 中求出最小容量者. 该容量 $u(f)$ 是待求最小截的容量.

另外，关于一般的梳子约束 (3.17) 以及团树约束 (3.20)，尚不知道有对生成边界面问题的多项式时间求解算法. 但是，有人提出了几个近似求解的多项式时间算法，被纳入分枝切割法. 因在分枝切割法中边界面的生成被多次迭代使用，所以特别要求提高它的效率. 为此，即使针对上面的排除子巡回路约束和 2- 匹配约束，在实用上也经常采用高速近似算法.

3.5 线性规划问题 $P(\mathcal{L}, F_0, F_1)$ 的处理

式 (3.14) 的线性规划问题 $P(\mathcal{L}, F_0, F_1)$，是有 $m = (n-1)/2$ 个变量 $x(e)$ 和 $n + |\mathcal{L}|$ 个约束条件 (而且还有 $x(e) \geq 0, x(e) \leq 1$ 型的约束) 的大规模问题. 为有效进行处理，人们在以下各方面下工夫，即让这些变量和约束条件的一部分呈显式而在有必要时生成其它部分.

3.5.1 变量的处理

根据报告的计算实验， $P(\mathcal{L}, F_0, F_1)$ 生成的边界面之中，成立等号者最多有 n 个. 于是，和约束条件 $Ax = 2$ 一起，可以认为实际有效的约束条件最多是 $2n$ 个. 也就是，该取正值的基底变量的个数最多是 $2n$ 个. 另外，根据约束条件 $Ax = 2$,

取值 $x(e)=1$ 的变量个数在 n 以下，所以满足 $x(e)=1$ 的非基变量（正确说来 $x(e)+t(e)=1$ 的松弛变量 $t(e)$ 是非基变量，即 $t(e)=0$）的个数也在 n 以下．于是，合起来，该取正值的变量个数最多是 $3n$ 个．剩下的 $m-3n$ 个变量取 0 值．

然而，因为事先并不知道哪个变量取正值，故采用其推测集合 $J\subseteq E$ 来使计算进行下去．计算开始时，注意算法 BCUT 的第一步中根据近似算法得到几个巡回路，令其中用到的边的集合为 J．把变量限定在 J 中，$P(\mathcal{L},F_0,F_1)$ 变成如下形式．

$$
\begin{aligned}
P_J(\mathcal{L},F_0,F_1)\quad &\text{目标函数：}\quad c_J^T x_J \quad\rightarrow\quad \text{最小}\\
&\text{约束条件：}\quad A_J x_J=2,\ 0\le x_J\le 1\\
&\qquad\qquad\qquad L_J x_J\le l^0 \qquad\qquad\qquad (3.37)\\
&\qquad\qquad\qquad x(e)=0\quad (e\in F_0\cap J)\\
&\qquad\qquad\qquad x(e)=1\quad (e\in F_1\cap J)
\end{aligned}
$$

这里，x_J 是把变量向量 x 限定在 $e\in J$ 后的结果，$Lx\le l^0$ 是把约束条件集合 $l^Tx<l_0$ $((l,l_0)\in\mathcal{L})$ 写成矩阵形式后的结果．c_J，A_J，L_J 分别是从 c，A，L 之中仅取出与变量 $x(e)$ $(e\in J)$ 对应的分量后的结果．

$P_J(\mathcal{L},F_0,F_1)$ 的最优解 $\overline{x}_J$ 是 $P(\mathcal{L},F_0,F_1)$ 的可行解（关于 J 以外的变量，令 $x(e)=0$ $(e\in E-J)$），但并不一定是最优解，所以线性规划中采用被称为**列生成法** (column generation method) 的下述迭代过程．

过程 CG (根据列生成法的 $P(\mathcal{L},F_0,F_1)$ 解法)

第一步 (初始化)： 从适当的 $J\subseteq E$ 开始．这里，要求选取 J 使得图 $G_J=(V,J)$ 至少包含一个巡回路．

第二步 (LP 解): 求出 $P(\mathcal{L}, F_0, F_1)$ 的最优解 $\bar{x}_J$ 同时也求出针对约束条件 $A_J x_J = 2$ 以及 $L_J x_J \leq l^0$ 的单纯形乘子向量 (参照 1.3 节和 1.7 节) u 和 $\bar{v}$

第三步 (最优性判定): 求出 J 以外的变量的相对成本 (参照 1.3 节)

$$r(e) = c(e) - u^T A_e - \bar{v}^T L_e \quad (e \in E - J) \tag{3.38}$$

这里, A_e (L_e) 表示 A (L) 的变量 $x(e)$ 所对应的列. 如果对所有的 $e \in E - J$ 都有 $r(e) \geq 0$, 则结束计算. $\bar{x}_J$ 是 $P(\mathcal{L}, F_0, F_1)$ 的最优解. 否则进入第四步.

第四步 (更新): 令

$$J := J \cup \{e \in E - J | r(e) < 0\}$$

回到第二步

通常如果考虑到 $|J| \ll |E|$, 则第三步很费时间. 分枝切割法里, 对同样的 F_0 和 F_1 更新 $\mathcal{L}$ 时多次求解 $P(\mathcal{L}, F_0, F_1)$, 每次都起动过程 CG 并不见得合算. 这是因为对现在的 J 解 $P_J(\mathcal{L}, F_0, F_1)$ 得到的 x_J 即使不是最优, 只要能够生成边界面, 分枝切割法就可以照样进行下去. 作为现实的妥协方案, 可以考虑仅在下述条件之一成立时起动过程 CG 的规则:

- $\mathcal{L} = \emptyset$ 的时候.
- 基于 $\bar{x}_J$ 生成边界面时失败了的时候.
- 多次 (比如 5 次) 没有采用过程 CG 解 $P_J(\mathcal{L}, F_0, F_1)$ 的时候.

3.5.2 约束条件的处理

得到 $P(\mathcal{L}, F_0, F_1)$ 的最优解 $\bar{x}$ 的时候、把约束条件 $l^T x < l_0$ ($(l, l_0) \in \mathcal{L}$) 中成立等号的集合记为 $\mathcal{L}_a$. $P(\mathcal{L}_a, F_0, F_1)$ 也同样具有最优解 x, 故属于 $\mathcal{L} - \mathcal{L}_a$ 的约束条件从这个时候开始就成为多余的了. 因此、 $P(\mathcal{L}, F_0, F_1)$ 的计算结束时, 把 $\mathcal{L} - \mathcal{L}_a$ 的约束条件转移到存放处 $\overline{\mathcal{L}}$, 从而把问题缩小成为 $P(\mathcal{L}_a, F_0, F_1)$ 后进行计算 (生成新的边界面等等) 这不是单纯的舍去, 之所以留在存放处 $\mathcal{L}$ 是因为它们很有可能给出其它 $\mathcal{L}'$ 的 $P(\mathcal{L}', F_0, F_1)$ 的有效边界面. 因此、算法 BCUT 的第五步 (生成边界面) 中首先查一查 $\mathcal{L}$ 里有无可以利用的约束, 仅在没有的情况下起动边界面的生成过程.

关于存放处 $\overline{\mathcal{L}}$ 的大小, 如果不加处理、可能漫无边际地增长下去, 所以有必要在适当的时期加以整理, 使之恒保持适当的大小.

3.5.3 变量的固定

下面下的工夫是, 利用初始问题 $P(\mathcal{L}, \emptyset, \emptyset)$ 的信息, 预先固定 (fixing) 一部分变量为 0 或者 1. 假定给定了暂定值 z^*, $P(\mathcal{L}, \emptyset, \emptyset)$ 的最优解 $\bar{x}$ 及其值 $\bar{z} = c^T \bar{x}$, 变量 $x(e)$ 的相对成本 $r(e)$ ($e \in E$) (参照式 (3.38)). 这时, 由相对成本表达的目标函数值的式 (1.10) 成为

$$z = \bar{z} + \sum_{e \in E} r(e) x(e)$$

(这里, 关于基底变量 $x(e)$ 有 $r(e) = 0$) 于是, 注意到 $r(e) \geq 0$ (根据 $\bar{x}$ 的最优性质) 和 $x(e) \geq 0$, 把上式限定于一个成立 $\bar{x}(e) = 0$ 的变量 $x(e)$, 写下来就得到

$$z \geq \bar{z} + r(e) x(e)$$

其结果，把这个变量固定在 $x(e)=1$ 后的 LP 问题的最优解 z 满足 $z \geq \bar{z}+r(e)$，所以如果

$$\bar{z}+r(e) \geq z^* \quad \text{也即} \quad r(e) \geq z^* - \bar{z} \tag{3.39}$$

则让 $x(e)=1$ 不可能得到比暂定值 z^* 更好的值．于是，在这种情况下，即使固定 $x(e)=0$ 也不会失去一般性．

同样，在 $\bar{x}(e)=1$ 的时候，如果

$$r(e) < -(z^* - \bar{z}) \quad (<0) \tag{3.40}$$

则可以固定 $x(e)=1$．

根据计算实验，以这种方式固定下的变量占全变量的 70%，多的时候达 95% 以上．被固定下来的变量在以后的计算中没有考虑的必要，所以在提高计算效率方面非常有效

分枝切割法里，与上面的固定相区别，将属于 F_0 或者 F_1 的取值 $x(e)=0$ 或者 $x(e)=1$ 的变量称为**设定** (setting) **变量**．固定变量的效果波及所有的子问题 $P(\mathcal{L},F_0,F_1)$，但设定变量的效果仅仅影响到其子问题．各 $P(\mathcal{L},F_0,F_1)$ 里，尽管固定后的变量和设定后的变量都存在，其结果有可能进一步设定变量．比如，如果固定或者设定与连接某个节点 v 的两条边 e 和 e' 为 $x(e)=x(e')=1$，则根据条件 $Ax=2$，与 v 连接的其它边 e'' 可设定为 $x(e'')=0$

分枝切割法的计算中，在更新了暂定解 x^* 的情况下，如果对新的 x^* 检验式 (3.39)，(3.40)，则有进一步固定新的变量的可能性．于是，考虑这些检验与所需要的时间及效果之间的平衡，进行适宜的检验才是有效的．

3.6 分枝切割法的构成

本节叙述有关 3.2.3 的分枝切割法 BCUT 的剩余部分，也就是第二步的搜索法和第八步的分枝操作. 为使分枝切割法作为程序得以实现，还要求对数据结构等做细致入微的考察. 特别是，随着计算进行下去，如何存储所生成的子问题 $< F_0, F_1 >$ 的有关数据 (是包含有关其上一级的线性规划问题 $P(\mathcal{L}, F_0', F_1')$ 的数据好呢，还是仅仅存储有关 F_0, F_1 的最小限的数据好呢), 不同的存储方案对所需要的空间以及计算时间会产生巨大的差异. 但是，由于其详细讨论超出本书范围，读者可参考有关的文献.

3.6.1 搜索法

第二步里，从清单 S 之中选取一个下一步检验的子问题 $< F_0, F_1 >$. 原则上，选取任意一个 $< F_0, F_1 >$ 计算都是正确的，但是进行计算的状况要发生很大的变化. 该选定法决定以何种顺序考察图 3.4 那样的树的节点，所以称为**搜索法** (search strategy). 具有代表性的搜索法有以下几种：

1. 从最新生成的 $< F_0, F_1 >$ 中选取.

2. 着眼于生成 $< F_0, F_1 >$ 的 $P(\mathcal{L}, F_0', F_1')$, 选取使其最优解 $\bar{x}$ 的值 $c^T\bar{x}$ 达到最小者.

3. 用某种方法推断从 $< F_0, F_1 >$ 得到的最短巡回路的长度，选取具有最小推断值者.

搜索法 1 称为**深度优先搜索** (depth-first search). 这里，因为分枝操作 (第八步) 里最新生成的子问题有两个，必须事先决定

其中哪个优先．这个搜索法具有可以将 S 中存放的 $< F_0, F_1 >$ 的个数控制在 $m+1$ 个以下的特征，在存储空间方面是有利的．另外，与其它搜索法相比，程序比较简单也是特征之一．

如果采用以到结束计算为止被检验的子问题 $< F_0, F_1 >$ 的个数来评价计算效率的观点，则搜索法 2 被认为是有利的，因为检验的个数可以控制得比较少．由于利用了从 $P(\mathcal{L}, F_0', F_1')$ 得到的最短巡回路的下界值，在这个意义上把它称为**最好下界搜索** (best-bound search)．基于最小的下界值的理由如下：

- 关于选取的 $< F_0, F_1 >$，$P(\mathcal{L}, F_0, F_1)$ 的值有变小的倾向，作为其结果，更快找到短巡回路的可能性很大．
- 在 $P(\mathcal{L}, F_0, F_1)$ 不给出巡回路的情况下 (即，进入第八步的分枝操作)，即使这个 $< F_0, F_1 >$ 在后面选取，(因为 $P(\mathcal{L}, F_0, F_1)$ 的值很小) 根据第四步的下界值检验而终止的可能性很小，仍然进入第八步，因而并不有利．

因此，旅行商问题的分枝切割法里，采用最好下界搜索的情况好像很多．

搜索法 3 因为是启发式地求出推断值故称为**启发式搜索法** (heuristic search)．如果推断值的精度很高，则有合并搜索法 1 和 2 的优点的倾向，故很有效．但是，在旅行商问题的情况下，很难通过简单的方法得到好的推断值，故还没有被实用化．另外，如果采用上一级 $P(\mathcal{L}, F_0', F_1')$ 的值作为推断值，则与最好下界搜索法等价 (也就是，启发式搜索法作为特别情况包含有最好搜索法)．有时也把搜索法 2 和 3 合起来称为**最好优先搜索** (best-first search)．

3.6.2 分枝操作

第八步的分枝操作是根据用到的分枝变量 $e \in E$ 来决定的. 选取哪个变量更有效, 没有明确的判断标准. 但是直观上认为, 先选取对其后的计算有很大影响的变量要好些. 具体来说, 希望分枝操作结果得到的 2 个子问题 $< F_0, F_1 \cup \{e\} >$ 和 $< F_0 \cup \{e\}, F_1 >$ 至少有一方的值比起 $< F_0, F_1 >$ 的值有很大的上升. 因为这种子问题满足第四步的下界值检验条件因而被排除的可能性很大.

作为具有这种性质的变量 e 的条件, 可以考虑

- $P(\mathcal{L}, F_0, F_1)$ 的最优值 $\bar{x}$ 处, $\bar{x}(e)$ 的值离中间值 0.5 很近者;
- 目标函数的系数 $c(e)$ 很大者.

3.7 旅行商问题的计算实验

在根据分枝切割法的计算结果中, 介绍 Padberg 和 Rinaldi [PR 91] 的一部分. 这是报告了最好结果的论文之一.

表 3.1 从已报告的众多例子之中选取 5 个并概括它们的计算数据. 不论在哪个例子中, 平面上两点 i 和 j (分别由 x 坐标和 y 坐标给出) 的距离 $c(i,j)$ 是由下式给出的欧几里得距离 (四舍五入的结果)

$$c(i,j) = \left\lfloor \sqrt{(x(i) - x(j))^2 + (y(i) - y(j))^2} + \frac{1}{2} \right\rfloor$$

所生成的. 最大的 TH2392 里, 如果所有的变量全部以显式处理, 就成为包含有约 286 万个变量的大规模的问题.

表的第二段里记载了最短巡回路长 OPT 以及分枝切割法得到的几个数据. RATIO 表示的是以 $P(\emptyset,\emptyset,\emptyset)$ 的值为基准, $P(\mathcal{L},\emptyset,\emptyset)$ 的值(算法 BCUT 的第三步结束时的 c^Tx) 和 OPT 的比的百分数表示. 对特大规模的例子, 这个值接近 100 表明了边界面约束的效果.

表 3.1 对旅行商问题的分枝切割法的计算结果

	问题实例				
	GH096	TK136	ATT532	TK1002	TH2392
节点数 n	96	136	532	1002	2392
变量个数 m	4560	9180	141246	501501	2859636
$P(\mathcal{L},\emptyset,\emptyset)$ 的值 LB0	55019.8	96546 0	27655 6	258856.6	377986 5
RATIO[a]	92.4	97.0	97.1	99 0	99.8
近似巡回路长 UB	55733	96998	27985	266437	390474
最短巡回路长 OPT	55209	96772	27686	259045	378032
线性规划问题的总数	126	110	1098	334	112
列数 $\|J\|$ 的最大值	222	276	2278	2365	5273
生成的边界面约束的个数	553	788	16987	11020	10238
约束条件数的最大值	187	259	2273	2059	3908
生成子问题的个数	6	10	106	18	2
最短巡回路的计算时间 [b]	9	9	109	323	1534
$P(\mathcal{L},\emptyset,\emptyset)$ 的处理时间 [b]	19	31	547	1120	5465
线性规划问题的处理时间 [b]	24	27	8731	5225	5890
全计算时间 [b]	63	71	19805	11714	9515

a RATIO=100(LB0−LP1)/(OPT LP1) 这里, LP1 是 $P(\emptyset\ \emptyset,\emptyset)$ 的值

b 单位是秒, 使用的计算机是 IBM 3090/600

第三段落里记载了实际生成的边界面约束的个数和解出的线性规划问题 $P(\mathcal{L},F_0,F_1)$ 的总数. 这些线性规划问题的规模可以用显式使用的变量的个数(式 (3.37) 的 J) 以及必要的约束条件数(条件 $Ax=2$ 的个数 n 和边界面约束的总数) 来评

价，它们之中的最大值和变量的个数 m 与边界面的总数比起来小得多，表明 3.5 节中的各种努力见效了.

像表的第四段落里写的那样，分枝切割法生成的子问题 $< F_0, F_1 >$ 的个数几乎都小到让人感到吃惊. 其理由可以认为是

- 由于边界面约束的效果，各 $P(\mathcal{L}. F_0, F_1)$ 给出高精度的下界值，
- 对旅行商问题，有像 3.2.3 所提及的 Lin 和 Kernighan 的方法那样的非常好的近似解法，
- 分枝定界法的各要素 (搜索法，分枝变量的选定法等等) 运行良好，

等等.

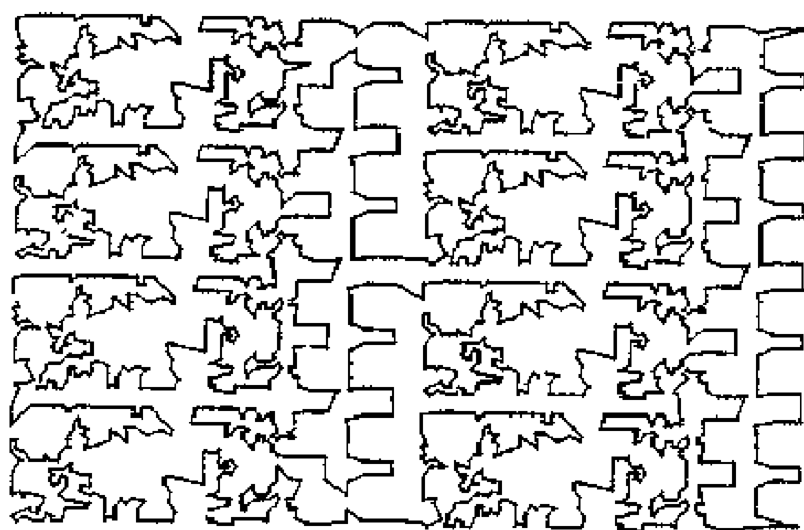


图 3.10 问题实例 TH 2392 的最短巡回路 [PR91]

表的最后的段落是关于计算时间的数据. 使用超大型计算

机 IBM 3090/600 时，解大规模的问题需要几小时．采用的单纯形法的软件包是 IBM 开发的 OSL 所需时间并不仅取决于节点数 n，可以看出有可能 n 小反而更消耗时间．最后，图 3.10 示意对应 $n = 2392$ 的问题实例的最短巡回路．

根据分枝切割法的计算实验，其后通过对算法的细微部分进行加工，在各处有了进展，最近有报告说解出了 $n > 3000$ 的问题例子．

3.8 文献及其它话题

对于包含旅行商问题的各种各样的最优化问题（特别是 NP 困难者），基于分枝定界法算法的开发很盛行，取得了一定的成功．结果，在分枝切割法里，根据多面体方法可得到高精度下界值，使得实用上的界限向前迈进了一步．

整数多面体的理论是以 1935 年的 Weyl 的定理为开端，但是关于旅行商问题，是以 1954 年的 Dantzig, Fulkerson, Johnson [DF 54] 提出的方案开始的．另外，该论文也是有系统地构造分枝定界法的最早的论文之一．其后，接连发现了旅行商问题的边界面约束（[GP 79] [GP 86] 等），报告利用这些结果的分枝切割法得出很好的计算结果的同时，也起到了多面体方法的牵引车的作用．关于边界面约束，除了 3.3 节所述的梳子约束，2- 匹配约束，团约束之外，还有路树 (path-tree) 约束和 k- 路 (k-path) 约束等等．进一步，还有不限于边界面有效约束，双巢 (binested) 约束和二分 (bipartition) 约束等为数众多的类型 [CFN 85] [BC 91] [NR 91] 在 [GH 91] [PR 91] 里报告有包含大规模问题实例的数值实验．本章的内容，包括 3.7 的计算实验，

主要根据 [PR 91].

旅行商问题本身是很有趣的研究对象，长久以来从各个侧面进行了研究. [LLRS 85] 是综述这些成果的书. 关于近似解法，该书也叙述了有代表性的 Lin, Kernighan [LK 73] 的结果. 关于分枝定界法的一般叙述，以及在其它问题中的应用等等，[I 83] 都有很详细的论述.

3.4 节利用了与边界面约束的生成相关联的网络流上的各种成果. 该领域的主要文献已经在 2.6 节给出. 求图的最小截算法里有 3.4 节提及的 Gomory, Hu [GH 61] 及其改良版本 [PR 90], 还有与最大流问题无关的 [NI92] 等新方法.

关于旅行商问题的变形也有广泛的研究. 比如，利用多台车访问全部节点的问题，考虑了经过各点时要卸下货物，同时也考虑了车子的容量限制等等问题. 关于这些问题的展开请参看 [LLRS 85] 等文献.

第 4 章 非线性最优化

本章讨论非线性最优化问题. 首先说明针对无约束问题的有代表性的方法——梯度法、牛顿法、拟牛顿法以及共轭梯度法的基本思想和性质. 其次叙述针对一般带约束问题的最优条件, 讲解带约束最优化里最有效的一个方法, 逐次二次规划法(SQP 法). 最后, 叙述基于凸分析的强有力方法, 接近点法.

4.1 无约束最优化问题

本章前半部分考察没有约束条件的最优化问题.

$$\text{目标函数:}\ f(x) \longrightarrow \text{最小} \tag{4.1}$$

这里, 变量 x 是 n 维实向量, 目标函数 $f: R^n \to R$ 假定为二次连续可微. 另外, 向量全为列向量, 即 $x = (x_1, x_2, \cdots, x_n)^T$ (T 为转置记号).

可以认为非线性函数一般像图 4.1 那样有几个山峰和山谷. 这里, 图 4.1 中点 a 是在变量 x 的全领域里使目标函数 f 达最小的点, 这样的点称为问题 (4.1) 的 **全局最优解** (global optimal solution); 点 d 仅限在它的适当邻域里使目标函数达最小, 这样的点称为 **局部最优解** (local optimal solution). 当然全局最优解也是局部最优解.

特别是, 人们知道, 目标函数 f 为 **凸函数** (convex function)

的时候（参照附录 A 2），问题 (4.1) 没有像图 4.1 的点 d 那样（不是全局最优解）的局部最优解．但是，事先知道给定的目标函数为凸函数的情况很少，一般不得不致力于存在多个局部最优解的问题．在一般情况下，从非特定的多个局部最优解中，找出全局最优解实际上非常困难．因此，通常的非线性最优化里，认为局部最优解就是要求的解，也就是把如何有效地找出局部最优解作为中心课题．

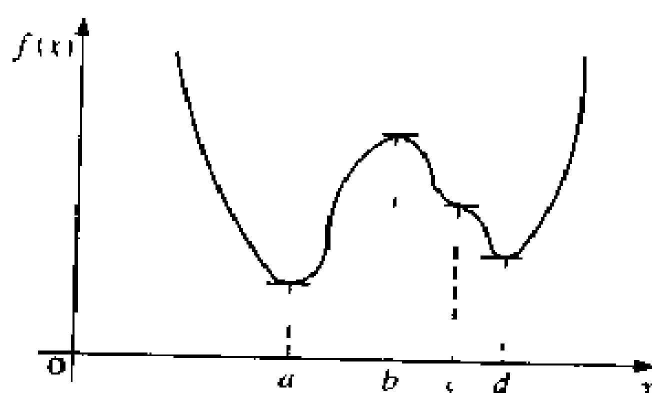


图 4.1 全局最优解和局部最优解

非线性最优化里函数的梯度这个概念起重要作用．函数 $f: R^n \to R$ 在点 $x \in R^n$ 的**梯度** (gradient)，是以函数 f 的偏微分系数 $\partial f(x)/\partial x_j\ (j = 1, 2, \ldots, n)$ 为分量的 n 维向量，用

$$\nabla f(x) = (\partial f(x)/\partial x_1, \partial f(x)/\partial x_2, \ldots, \partial f(x)/\partial x_n)^T$$

表示．众所周知，如果点 x^* 是问题 (4.1) 的局部最优解，则成立

$$\nabla f(x^*) = 0 \tag{4.2}$$

但是，满足式 (4.2) 的点 x^*，像图 4.1 的点 b 和 c 那样，并不一定限于问题 (4.1) 的局部最优解．也即，一般说来，式 (4.2) 是点 x^* 成为问题 (4.1) 的局部最优解的必要条件而不是充分条件．当然，f 为凸函数的时候式 (4.2) 是点 x^* 成为问题 (4.1) 的全

局最优解的充分必要条件. 以下, 称满足式 (4.2) 的点为函数 f 的**稳定点** (stationary point).

进一步详细考查某个点是不是问题 (4.1) 的局部最优解的时候, 有必要考虑函数 f 的二次微分系数. n 个变量的函数 f 的二次微分系数可用以 $\partial^2 f(x)/\partial x_i \partial x_j$ 为第 i 行第 j 列元素的 $n \times n$ 矩阵来表示. 称该矩阵为函数 f 在点 x 的**海赛矩阵** (Hessian matrix), 用 $\nabla^2 f(x)$ 表示. 当函数 f 为二次连续可微时, 因为 $\partial^2 f(x)/\partial x_i \partial x_j = \partial^2 f(x)/\partial x_j \partial x_i$ 成立, 所以 $\nabla^2 f(x)$ 是对称矩阵.

如果点 x^* 是问题 (4.1) 的局部最优解, 则 f 在点 x^* 的海赛矩阵为半正定的, 也即

$$y^T \nabla^2 f(x^*) y \geq 0 \quad (\text{对任意的} y \in R^n) \tag{4.3}$$

成立. 另外, 当 x^* 是 f 的稳定点, 而且 f 在点 x^* 的海赛矩阵正定, 也就是

$$y^T \nabla^2 f(x^*) y > 0 \quad (\text{对任意的} y \in R^n,\ y \neq 0) \tag{4.4}$$

时, 也可以断言点 x^* 是问题 (4.1) 的局部最优解. 因此, 式 (4.2) 和式 (4.3) 合起来称为针对问题 (4.1) 的最优性的**二次必要条件** (second-order necessary condition), 式 (4.2) 和式 (4.4) 为**二次充分条件** (second-order sufficient condition).

4.2 梯度法

梯度向量 $\nabla f(x)$ 是函数 f 在点 x 的增加率达最大的方向, 故在最小化函数 f 时它成为最重要的工具. 实际上, 针对无约束最优化问题 (4.1), 到现在为止大家知道的解法中大多属于下面的称为**梯度法** (gradient method) 的方法类.

算法 GRADIENT (梯度法)

第一步 (初始化): 选取适当的初始点 $x^{(0)} \in R^n$, 令 $k := 0$

第二步 (搜索方向的计算): 利用适当的正定对称矩阵 $H^{(k)}$ 计算搜索方向向量 $d^{(k)} := -H^{(k)}\nabla f(x^{(k)})$. (如果 $\nabla f(x^{(k)}) = 0$ 则结束计算.)

第三步 (一维搜索): 解一维最优化问题

$$\min_{t>0} f(x^{(k)} + td^{(k)}) \tag{4.5}$$

求出步长 $t = t^{(k)}$, 令 $x^{(k+1)} := x^{(k)} + t^{(k)}d^{(k)}$, $k := k+1$ 回到第二步.

因为矩阵 $H^{(k)}$ 正定, 如果 $\nabla f(x^{(k)}) \neq 0$ 则成立

$$\nabla f(x^{(k)})^T d^{(k)} = -\nabla f(x^{(k)})^T H^{(k)} \nabla f(x^{(k)}) < 0$$

也就是, $d^{(k)}$ 是函数 f 在点 $x^{(k)}$ 的 **下降方向** (descent direction) 特别是, 如果让矩阵 $H^{(k)}$ 恒为单位矩阵 I 则 $d^{(k)} = -\nabla f(x^{(k)})$. 称该 $d^{(k)}$ 为最速下降方向. 此时上面的方法称为 **最速下降法** (steepest descent method)

实际计算中, 在第二步执行收敛判定. 比如, 对于事先给定的很小的正常数 ϵ, 如果

$$\|\nabla f(x^{(k)})\| < \epsilon$$

成立, 则结束计算 ($\|\cdot\|$ 是欧几里得范数).

解一维最优化问题 (4.5) 确定步长的方法称为 **一维搜索** (line search). 特别是, 上面的算法以问题 (4.5) 的最优解为步长, 这种方法有时也被称为 **精确一维搜索** (exact line search). 一维搜

索里经常用到的方法有**黄金分割搜索**(golden section search)和利用二次或三次**插值**(interpolation)的迭代法。但是，即使说是精确的一维搜索，通过有限次计算求出问题(4.5)的严密解一般也是不可能的，因此实际上在得到问题(4.5)的有足够精度的近似解时就采用它为步长.

另外，实际计算时不是解一维最优化问题(4.5)，而是经常找出满足某个适当条件，比如同时满足下面两个不等式的大于0的 t，把它当做步长 $t^{(k)}$.

$$\begin{cases} f(x^{(k)}+td^{(k)}) \leq f(x^{(k)}) + \beta t \nabla f(x^{(k)})^T d^{(k)} \\ \nabla f(x^{(k)}+td^{(k)})^T d^{(k)} \geq \gamma \nabla f(x^{(k)})^T d^{(k)} \end{cases} \tag{4.6}$$

这里，β 和 γ 是满足 $\beta \in (0,1/2)$ 及 $\gamma \in (\beta,1)$ 的常数. 现在，定义函数 $\varphi: R \to R$, $\varphi(t) = f(x^{(k)}+td^{(k)})$. 因为 $\varphi'(t) = \nabla f(x^{(k)}+td^{(k)})^T d^{(k)}$，条件(4.6)可以改写为

$$\begin{cases} \varphi(t) \leq \varphi(0) + \beta t \varphi'(0) \\ \varphi'(t) \geq \gamma \varphi'(0) \end{cases}$$

图4.2示意了满足条件(4.6)的 t 的范围. 从图4.2可以看出，求满足条件(4.6)的 t 对应于求出一维最优化问题(4.5)的粗略近似解，所以经常被称为**非精确一维搜索**(inexact line search). 采纳非精确一维搜索的结果，与精确一维搜索相比，虽然有些情况下梯度法的迭代次数要增加，但是每次一维搜索搜索所要的计算量少，所以特别是对于计算函数值很耗工夫的问题，从整体来说在很多情况下可以提高计算效率.

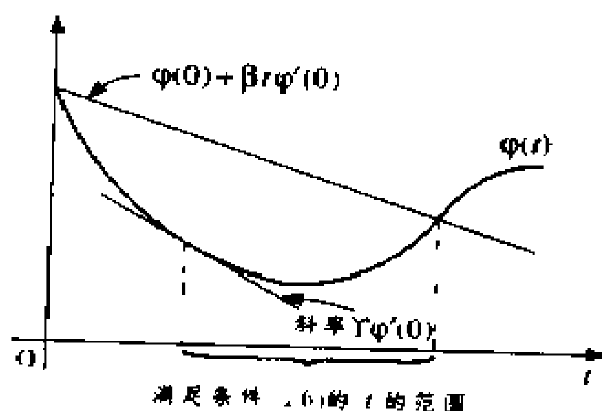


图 4.2 非精确一维搜索

如果某迭代法生成点列的极限 (聚点), 在适当假定下, 可以保证与初始点无关且恒为问题的最优解 (或者稳定点), 则称该迭代法有**全局收敛性**(global convergence property)[1]. 与此相反, 如果在解的附近选取初始点的话, 可以保证所生成的点列收敛于该解, 称这样的迭代法有**局部收敛性** (local convergence property). 关于上面所讲的梯度法的全局收敛性, 成立下面的定理.

定理 4.1 (梯度法的全局收敛性) 假定对于任意初始点 $x^{(0)} \in R^n$, 集合 $\{x \in R^n, f(x) < f(x^{(0)})\}$ 有界, 而且对 $n \times n$ 对称矩阵的列 $\{H^{(k)}\}$, 存在满足

$$\lambda y^T y \le y^T H^{(k)} y \le \Lambda y^T y \quad (y \in R^n,\ k = 0, 1, \ldots)$$

的常数 $\Lambda > \lambda > 0$. 这时, 梯度法算法所生成的点列 $\{x^{(k)}\}$ 的任意聚点是问题 (4.1) 的稳定点.

该定理讲的是梯度法全局收敛于 (比局部最优解更弱的概念) 稳定点. 但是在一般情况下, 不妨认为根据梯度法实际找

1) 注意全局收敛性并不是保证收敛于全局最优解的概念.

出的稳定点是问题的局部最优解. 然而, 问题有多个局部最优解时, 梯度法收敛于哪个解一般依赖于初始点的选取方法. 为此, 实际计算中, 用几个初始点进行计算, 在所得到的局部最优解中采用最好者作为问题的解. 这种做法很普遍.

其次, 考虑有关梯度法的收敛速度. 首先, 为简单起见, 假定目标函数为如下的凸二次函数:

$$f(x) = \frac{1}{2}x^T Bx - c^T x \tag{4.7}$$

这里, 函数 f 的海赛矩阵 B 为正定对称的. 显然, 问题 (4.1) 有唯一的全局最优解 $x^* = B^{-1}c$. 根据定理 4.1, 梯度法生成的点列收敛于 x^*. 另外, 关于其收敛速度, 知道有下面的结果.

定理 4.2 (梯度法的收敛速度) 若目标函数 f 是式 (4.7) 所定义的凸二次函数, 对于执行精确一维搜索的梯度法算法生成的点列 $\{x^{(k)}\}$ 成立下式:

$$f(x^{(k+1)}) - f(x^*) \leq \left(\frac{M^{(k)} - m^{(k)}}{M^{(k)} + m^{(k)}}\right)^2 \left\{f(x^{(k)}) - f(x^*)\right\}$$
$$(k = 0, 1, \ldots) \tag{4.8}$$

这里, $x^* = B^{-1}c$, $M^{(k)}$ 和 $m^{(k)}$ 分别是矩阵 $(H^{(k)})^{1/2}B(H^{(k)})^{1/2}$ 的最大和最小特征值[1]. 另外, 因矩阵 $(H^{(k)})^{1/2}B(H^{(k)})^{1/2}$ 是正定的, 故 $M^{(k)} \geq m^{(k)} > 0$.

该定理表达了, 在点 $x^{(k)}$ 的目标函数值 $f(x^{(k)})$ 和最小值 $f(x^*)$ 的差, 在每次迭代里按比率 $[(M^{(k)}-m^{(k)})/(M^{(k)}+m^{(k)})]^2 < 1$ 减少下去. 特别是, 在最速下降法里, 因为 $H^{(k)} = I$, $M^{(k)}, m^{(k)}$ 等于 (不依赖于迭代 k 的) 函数 f 的海赛矩阵 B 的最大及最小

---

1) 对于任意的正定对称矩阵 A, 存在有唯一的正定对称矩阵 C 满足 $A = CC$. 把该矩阵记为 $A^{1/2}$

特征值. 于是, 若令 B 的最大特征值为 M, 最小特征值为 m, 则式 (4.8) 变成如下.

$$\begin{aligned} & f(x^{(k+1)}) - f(x^*) \\ & \le \left(\frac{M-m}{M+m}\right)^2 \left\{f(x^{(k)}) - f(x^*)\right\} \\ & = \left(\frac{\tau-1}{\tau+1}\right)^2 \left\{f(x^{(k)}) - f(x^*)\right\} \qquad (k=0,1,\ldots) \end{aligned} \tag{4.9}$$

这里, $\tau = M/m\ (\ge 1)$ 一般称为矩阵 B 的**条件数** (condition number). 它是表达二次函数 (4.7) 性质的重要指标. 特别是, 根据式 (4.9), 可以认为条件数 τ 很小 (接近 1) 时, $f(x^{(k)})$ 快速收敛于最小值 $f(x^*)$. 相反, τ 很大时收敛很慢 (图 4.3).

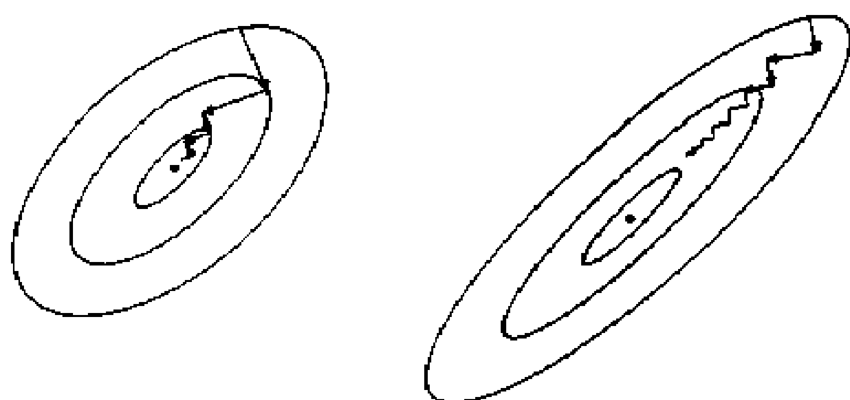


图 4.3 最速下降法的收敛速度和条件数

下面, 考虑点 $x^{(k)}$ 以怎样的速度收敛于最优解 x^*. 根据式 (4.9) 有

$$f(x^{(k)}) - f(x^*) \le \left(\frac{\tau-1}{\tau+1}\right)^{2k} \left\{f(x^{(0)}) - f(x^*)\right\} \quad (k=1,2,\ldots) \tag{4.10}$$

另一方面, 由二次函数的性质, 对矩阵 B 的最小特征值 m 成立有

$$f(x) - f(x^*) \ge \frac{m}{2}\|x - x^*\|^2 \quad (x \in R^n) \tag{4.11}$$

所以根据式 (4.10) 和 (4.11), 结果有

$$\|x^{(k)} - x^*\| \le c\left(\frac{\tau - 1}{\tau + 1}\right)^k \quad (k = 1, 2, \cdots) \tag{4.12}$$

(这里, $c = \sqrt{2\{f(x^{(0)}) - f(x^*)\}/m}$) 这个结果表明, 点列 $\{x^{(k)}\}$线性收敛(linear convergence)[1] 于 x^*.

上面考虑了针对二次函数的梯度法之收敛速度. 即使目标函数 f 为一般的非线性函数的时候也成立同样的结果. 特别是, 执行精确一维搜索的梯度法所生成的点列 $\{x^{(k)}\}$ 收敛于问题 (4.1) 的局部最优解 x^*, 海赛矩阵 $\nabla^2 f(x^*)$ 为正定的时候, 成立有

$$\limsup_{k\to\infty} \frac{f(x^{(k+1)}) - f(x^*)}{f(x^{(k)}) - f(x^*)} \le \limsup_{k\to\infty}\left(\frac{\hat{\tau}^{(k)} - 1}{\hat{\tau}^{(k)} + 1}\right)^2 \tag{4.13}$$

这里 $\hat{\tau}^{(k)}$ 是矩阵 $(H^{(k)})^{1/2}\nabla^2 f(x^*)(H^{(k)})^{1/2}$ 的条件数. 于是, 特别针对最速下降法, 若令在点 x^* 的海赛矩阵 $\nabla^2 f(x^*)$ 的条件数为 τ^*, 则成立

$$\limsup_{k\to\infty} \frac{f(x^{(k+1)}) - f(x^*)}{f(x^{(k)}) - f(x^*)} \le \left(\frac{\tau^* - 1}{\tau^* + 1}\right)^2 \quad (< 1) \tag{4.14}$$

也即, 最速下降法的收敛速度在很大程度上依赖于在最优解 x^* 处目标函数的海赛矩阵的条件数 τ^* 的值. 必须认识到, 因为在实际问题里多数情况下 τ^* 的值相当大, 若利用最速下降法解问题, 则需要相当多的迭代次数.

---

1) 一般, 如果存在常数 $c > 0$ 和 $0 < r < 1$ 当 k 足够大时成立 $\|x^{(k)} - x^*\| \le cr^k$ 的话, 就称点列 $\{x^{(k)}\}$ R-1 次收敛于 x^*, 如果成立 $\|x^{(k+1)} - x^*\| \le r\|x^{(k)} - x^*\|$ 的话, 就称为 Q-1 次收敛. Q-1 次收敛的点列一定 R-1 次收敛, 但反过来未必正确. 这里, Q 是 quotient, R 是 root 的意思. 在没有特地明确区别两者的必要时, 仅称 $\{x^{(k)}\}$ 一次收敛于 x^*, 或者 $\{x^{(k)}\}$ 的收敛率 (convergence rate) 是一次.

4.3 牛顿法

在梯度法的各次迭代中，若选取与 $\nabla^2 f(x^*)$ 的逆矩阵接近的矩阵作为 $H^{(k)}$，则矩阵 $(H^{(k)})^{1/2}\nabla^2 f(x^*)(H^{(k)})^{1/2}$ 近似等于单位矩阵．这时，该矩阵的条件数 $r^{(k)}$ 几乎等于 1，所以根据式(4.13)，可以期待点列 $\{x^{(k)}\}$ 非常快收敛．

特别地，在目标函数为凸二次函数

$$f(x) = \frac{1}{2}x^T Bx - c^T x \quad (B\text{为正定对称})$$

的情况下，对任意的 x 因为 $\nabla^2 f(x) = B$，在第一次迭代里令 $H^{(0)} = B^{-1}$，则

$$d^{(0)} = -H^{(0)}\nabla f(x^{(0)}) = -B^{-1}\left(Bx^{(0)} - c\right) = -(x^{(0)} - x^*)$$

这里，$x^* = B^{-1}c$ 是问题的最优解，$x^{(0)} \neq x^*$．这时如果记

$$\varphi(t) = f(x^{(0)} + td^{(0)}) = f(x^{(0)} - t(x^{(0)} - x^*))$$

则因为

$$\begin{aligned}
\varphi'(t) &= \nabla f\left(x^{(0)} - t(x^{(0)} - x^*)\right)^T \left(x^{(0)} - x^*\right) \\
&= \left(B\left(x^{(0)} - t(x^{(0)} - x^*) - x^*\right)\right)^T \left(x^{(0)} - x^*\right) \\
&= (1-t)\left(x^{(0)} - x^*\right)^T B\left(x^{(0)} - x^*\right)
\end{aligned}$$

根据精确的一维搜索条件 $\varphi'(t) = 0$，步长成为 $t^{(0)} = 1$．于是，

$$x^{(1)} = x^{(0)} + d^{(0)} = x^*$$

由此知道，不管初始点 $x^{(0)}$ 如何，在一次迭代后到达最优解 x^*．

根据这个事实，可以认为即使对于一般的非线性函数 f，各迭代里令

$$H^{(k)} = \nabla^2 f(x^{(k)})^{-1}$$

确定搜索方向

$$d^{(k)} = -\nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}) \tag{4.15}$$

是合适的．特别是，对于该搜索方向 $d^{(k)}$，步长恒为 1 的迭代法

$$x^{(k+1)} = x^{(k)} + d^{(k)} = x^{(k)} - \nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}) \tag{4.16}$$

称为**牛顿法** (Newton method)．由式 (4.15) 给出的搜索方向 $d^{(k)}$ 称为**牛顿方向**(Newton direction)．对牛顿法成立下面的定理．

定理 4.3 (牛顿法的局部收敛性和二次收敛性) 目标函数 f 的海赛矩阵为李普希兹连续[1)]，在问题 (4.1) 的局部最优解 x^* 处 $\nabla^2 f(x^*)$ 正定．这时，如果选取初始点 $x^{(0)}$ 与 x^* 充分接近，则牛顿法 (4.16) 所生成的点列 $x^{(k)}$ 收敛于 x^*．进一步，存在某个常数 $\eta > 0$，对所有的 k 成立

$$\|x^{(k+1)} - x^*\| \leq \eta \|x^{(k)} - x^*\|^2 \tag{4.17}$$

对足够大的任意 k，式 (4.17) 成立时，称 $x^{(k)}$ **二次收敛** (quadratic convergence) 于 x^*，也称收敛率为二次．因为二次收敛一般意味着收敛非常快，所以定理 4.3 表明牛顿法在收敛速度方面有非常出色的特性．

但是，式 (4.16) 的牛顿法里，如果不在解 x^* 的附近选取初始点 $x^{(0)}$，那么生成的点列 $x^{(k)}$ 未必收敛于最优解．为保证全

---

1) 所谓 $\nabla^2 f(x)$ 为**李普希兹连续** (Lipschitz continuous) 是说，对于某个常数 $L > 0$ 成立 $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|$，$(x, y \in R^n)$

局收敛性，有必要对牛顿法作某些修正．作为方法之一，可以考虑像下降法的情况一样导入一维搜索．然而，除去类似于函数 f 的海赛矩阵 $\nabla^2 f(x)$ 恒为正定的特别情况外，一般式 (4.15) 的牛顿方向 $d^{(k)}$ 未必成为 f 的下降方向，所以照搬原样使用一维搜索是不合适的

下面要讲的**信赖域法** (trust region method) 是不执行一维搜索而使牛顿法有全局收敛性的方法．它因在理论上出色而闻名．

首先注意，牛顿法的迭代 (4.16) 可以看成是先求得使 f 的二次逼近函数

$$F^{(k)}(d) = f(x^{(k)}) + \nabla f(x^{(k)})^T d + \frac{1}{2} d^T \nabla^2 f(x^{(k)}) d \tag{4.18}$$

在点 $x^{(k)}$ 达最小的 $d = d^{(k)}$，然后由 $x^{(k+1)} = x^{(k)} + d^{(k)}$ 确定下一步的点 (图 4.4 的虚线表示各次迭代里二次逼近函数 $F^{(k)}$ 的等高线)．这里，向量 d 表示从当前的点 $x^{(k)}$ 开始的位移．当然，为了让该方法有意义海赛矩阵 $\nabla^2 f(x^{(k)})$ 必须为正定的．

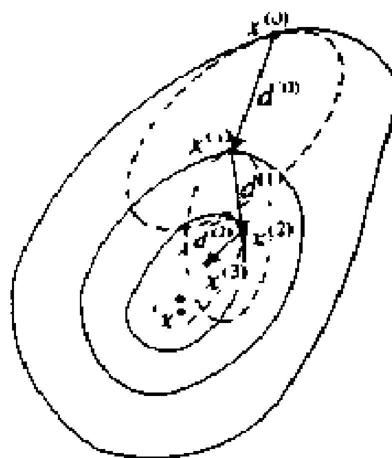


图 4.4 牛顿法的收敛

但是，在 f 为一般的非线性函数的情况下，式 (4.18) 的二

次函数 $F^{(k)}$ 仅在点 $x^{(k)}$ 的邻域里成为 f 的近似才有意义．于是，可以认为，即使使用牛顿法，也要限定在点 $x^{(k)}$ 的适当的邻域里，利用在该邻域里使函数 $F^{(k)}$ 最小的 $d = d^{(k)}$ 确定下一步的点 $x^{(k+1)}$．也即，对某个常数 $h^{(k)} > 0$，解二次函数的最小化问题

$$\begin{aligned}&\text{目标函数：} F^{(k)}(d) \longrightarrow \text{最小} \\ &\text{约束条件：} \|d\| < h^{(k)}\end{aligned} \tag{4.19}$$

利用其解 $d^{(k)}$ 确定下一步的点

$$x^{(k+1)} := x^{(k)} + d^{(k)}$$

这里，$\|d\| = \sqrt{d^T d}$．称该问题的可行域为**信赖域** (trust region)．因为它是有界闭集合，即使海赛矩阵 $\nabla^2 f(x^{(k)})$ 不是正定的，问题 (4.19) 也有解 $d^{(k)}$．决定信赖域大小的 h，若太大则问题 (4.19) 不能说成是对原来问题的适当逼近，相反若太小则 $d^{(k)}$ 比需要的更小、会增加收敛所需要的迭代次数．于是，有必要在各次迭代里，一边检查当前的信赖域半径 $h^{(k)}$ 的大小是否适当，一边执行计算．具体地，可以考虑如下算法．

算法 TR (信赖域法)

第一步 (初始化)： 选取常数 $0 < \mu_1 < \mu_2 < 1$, $0 < \gamma_1 < 1 < \gamma_2$ (比如，$\mu_1 = 0.2, \mu_2 = 0.8, \gamma_1 = 0.5, \gamma_2 = 2$)．选取适当的初始点 $x^{(0)} \in R^n$ 和正数 $h^{(0)}$，令 $k := 0$.

第二步 (位移向量的计算)： 令 $h = h^{(k)}$，解问题 (4.19) 求出向量 $d^{(k)}$．(如果 $d^{(k)} = 0$ 则结束计算．)

第三步 (迭代点的更新)： 令 $\Delta f^{(k)} = f(x^{(k)}) - f(x^{(k)} + d^{(k)})$，$\Delta F^{(k)}$

$:= f(x^{(k)}) - F^{(k)}(d^{(k)})$, 计算 $r^{(k)} := \Delta f^{(k)}/\Delta F^{(k)}$. 如果 $r^{(k)} \geq \mu_1$ 则 $x^{(k+1)} := x^{(k)} + d^{(k)}$, 如果 $r^{(k)} < \mu_1$ 则 $x^{(k+1)} := x^{(k)}$.

第四步 (信赖域的调整): 如果 $r^{(k)} \geq \mu_2$ 则 $h^{(k+1)} := \gamma_2 h^{(k)}$, 如果 $\mu_1 < r^{(k)} < \mu_2$ 则 $h^{(k+1)} := h^{(k)}$, 如果 $r^{(k)} < \mu_1$ 则 $h^{(k+1)} := \gamma_1 h^{(k)}$. 令 $k = k + 1$ 回到第二步.

第三步中计算的 $\Delta f^{(k)}$ 是点从 $x^{(k)}$ 移动到 $x^{(k)} + d^{(k)}$ 后目标函数值的减少量, $\Delta F^{(k)}$ 可看成是基于二次逼近函数 $F^{(k)}$ 所推定的 $\Delta f^{(k)}$ 的值. 特别是, 因为 $d^{(k)}$ 是问题 (4.19) 的解, 如果 $d^{(k)} \neq 0$ 则 $F^{(k)}(d^{(k)}) < F^{(k)}(0) = f(x^{(k)})$, 也即 $\Delta F^{(k)} > 0$. 第三步里的更新 $x^{(k+1)} := x^{(k)} + d^{(k)}$ 仅限于 $r^{(k)} = \Delta f^{(k)}/\Delta F^{(k)} > \mu_1 > 0$, 也即 $\Delta f^{(k)} > 0$ 的时候, 所以对应于生成的点列 $\{x^{(k)}\}$ 的目标函数值 $\{f(x^{(k)})\}$ 是单调非增加的.

上面的算法 TR 的基本思想可说明如下. 信赖域的半径 $h^{(k)}$ 的值如果设定得好的话, 可以使信赖域上逼近函数 $F^{(k)}$ 和真正的目标函数 f 的差很小. 这时问题 (4.19) 的解 $d^{(k)}$ 所确定的点 $x^{(k)} + d^{(k)}$ 可看成是该领域内函数 f 的最小点的一个很好的近似. 半径 $h^{(k)}$ 的值是否适当, 其判定可以基于第三步所定义的 $r^{(k)}$, 按如下方式进行. 首先在第三步里, 仅在目标函数值减少得足够多时 (即 $r^{(k)} \geq \mu_1$) 时从点 $x^{(k)}$ 移动到 $x^{(k)} + d^{(k)}$, 减少不多或者反而增加时 ($r^{(k)} < \mu_1$), 下次迭代便停在同一点. 在后一种情况下, 为了增加下次迭代里目标函数减少的可能性, 在第四步里缩小信赖域的半径. 在前一种情况下, 信赖域上逼近函数 $F^{(k)}$ 和真正的目标函数 f 的差可以认为非常小时 ($r^{(k)} \geq \mu_2$), 为了在下次迭代里点的移动能大一些从而导致加速收敛, 需要加大信赖域的半径.

定理 4.4 (信赖域法的全局收敛性和二次收敛性) 由信赖

域法生成的点列 $\{x^{(k)}\}$ 如果有界，则 $\{x^{(k)}\}$ 必存在聚点 x^* 满足问题 (4.1) 的最优性二次必要条件 (4.2), (4.3). 进一步，如果在该聚点 x^* 处最优性二次充分条件 (4.4) 成立，则 $\{x^{(k)}\}$ 二次收敛于 x^*.

执行一维搜索的方法所生成点列的聚点，一般理论上仅能保证是问题 (4.1) 的稳定点 (满足最优性的一次必要条件) (定理 4.1). 与此相比，该定理表示信赖域法有非常强的全局收敛性. 虽然信赖域法在理论上有这样出色的特性，但是它的实际有效性取决于各次迭代里能否有效地解出问题 (4.19). 由问题 (4.19) 的特殊结构可以断言，$d^{(k)}$ 是问题 (4.19) 全局最优解的充分必要条件是，存在有满足下面三个条件[1]的实数 $\lambda \ge 0$

$$\left(\nabla^2 f(x^{(k)}) + \lambda I\right) d^{(k)} = -\nabla f(x^{(k)}) \tag{4.20}$$

$$\|d^{(k)}\| \le h^{(k)},\ \lambda\left(\|d^{(k)}\| - h^{(k)}\right) = 0 \tag{4.21}$$

$$\nabla^2 f(x^{(k)}) + \lambda I \text{ 为半正定} \tag{4.22}$$

要找出满足条件 (4.20)-(4.22) 的 $d^{(k)}$ 和 $\lambda \ge 0$, 只要作如下处理即可. 首先，固定 λ, 考虑关于变量 d 的联立一次方程组

$$\left(\nabla^2 f(x^{(k)}) + \lambda I\right) d = -\nabla f(x^{(k)}) \tag{4.23}$$

把它的解表示为 $d(\lambda)$ (也即，把 $d(\lambda)$ 看成 λ 的函数 $d(\cdot): R \to R^n$). $d^{(k)}$ 的计算可分成下面的两阶段进行.

1 检查条件 (4.20)-(4.22) 对 $\lambda = 0$ 是否满足. 也即，$\nabla^2 f(x^{(k)})$ 是否正定，而且是否成立 $\|d(0)\| \le h^{(k)}$. 如果成立，则 $d^{(k)} = d(0)$ 和 $\lambda = 0$ 满足式 (4.20)-(4.22).

---

1) 条件 (4.20)-(4.22) 与 4.6 节要叙述的针对带约束最优化问题的最优条件相关联. 尽管所考虑的问题 (4.19) 不限于凸规划问题，前者的条件给出了最优性的充分必要条件，故它不能从后者的条件直接推出.

2. 否则，在成立式 (4.22) 的范围内[1] 求出满足

$$\|d(\lambda)\| = h^{(k)} \tag{4.24}$$

的 $\lambda > 0$. 计算这样的 λ 时，把式 (4.24) 看成是关于 λ 的非线性方程，用适当的迭代法解即可. 这样得到的 λ 所对应的 $d(\lambda)$ 是所要求的 $d^{(k)}$.

上面所讲的方法，作为确定信赖域的范数用了欧几里得范数. 也可采用其它的范数，比如，$\|d\|_\infty = \max_{1\le i\le n} |d_i|$. 这时，问题 (4.19) 成为在各变量的上下限约束 $-h^{(k)} \le d_i \le h^{(k)}$ $(i = 1,2,\dots,n)$ 下，最小化二次函数 $F^{(k)}(d)$ 的问题. 最后，请注意，用了欧几里得范数的信赖域法可以看成是非线性最小二乘问题的强有力的解法——Levenberg-Marquardt 方法的推广.

4.4 拟牛顿法

拟牛顿法 (quasi-Newton methods) 是 4.2 节所述梯度法的一种. 它在各次迭代中决定搜索方向时，选取矩阵 $H^{(k)}$ 为目标函数 f 的海赛矩阵 $\nabla f^2(x^{(k)})$ 的逆矩阵之近似，是利用有关 f 的梯度的信息逐次更新 $H^{(k)}$ 的方法.

算法 QN (拟牛顿法)

第一步 (初始化): 选取适当的初始点 $x^{(0)} \in R^n$. 令 $H^{(0)} = I$, $k := 0$.

第二步 (搜索方向的计算): 计算搜索方向向量

1) 在解联立一次方程组 (4.23) 计算 $d(\lambda)$ 时，如果用 Cholesky 分解，则可同时判定条件 (4.22).

$d^{(k)} := -H^{(k)}\nabla f(x^{(k)})$. (如果 $\nabla f(x^{(k)}) = 0$ 则结束计算.)

第三步 (一维搜索): 解一维最优化问题

$$\min_{t>0} f(x^{(k)} + td^{(k)}) \tag{4.25}$$

求出步长 $t = t^{(k)}$, $x^{(k+1)} := x^{(k)} + t^{(k)}d^{(k)}$.

第四步 (矩阵 $H^{(k)}$ 的更新): 基于点 $x^{(k)}$ 到 $x^{(k+1)}$ 的梯度向量的变化, 更新矩阵 $H^{(k)}$, 决定 $H^{(k+1)}$. 令 $k := k+1$ 回到第二步.

该方法的要点是第四步中矩阵 $H^{(k)}$ 的更新. 迄今为止, 提出了好几个更新 $H^{(k)}$ 的方法. 下面给出的两个更新公式是其中最基本的, 实际上用起来它们被认为是很有效的.

Davidon-Fletcher-Powell (DFP) 公式

$$H^{(k+1)} := H^{(k)} + \frac{s^{(k)}(s^{(k)})^T}{(s^{(k)})^T y^{(k)}} - \frac{H^{(k)}y^{(k)}(y^{(k)})^T H^{(k)}}{(y^{(k)})^T H^{(k)} y^{(k)}} \tag{4.26}$$

Broyden-Fletcher-Goldfarb-Shanno (BFGS) 公式:

$$H^{(k+1)} := H^{(k)} + \left(1 + \frac{(y^{(k)})^T H^{(k)} y^{(k)}}{(s^{(k)})^T y^{(k)}}\right) \frac{s^{(k)}(s^{(k)})^T}{(s^{(k)})^T y^{(k)}} - \left(\frac{s^{(k)}(y^{(k)})^T H^{(k)} + H^{(k)} y^{(k)} (s^{(k)})^T}{(s^{(k)})^T y^{(k)}}\right) \tag{4.27}$$

这里, $s^{(k)} := x^{(k+1)} - x^{(k)}$, $y^{(k)} := \nabla f(x^{(k+1)}) - \nabla f(x^{(k)})$. 如果

$$(s^{(k)})^T y^{(k)} > 0 \tag{4.28}$$

则根据这些更新公式确定的矩阵 $H^{(k)}$ 有

$$H^{(k)} \text{ 正定对称} \Longrightarrow H^{(k+1)} \text{ 正定对称} \tag{4.29}$$

的性质．另外，在拟牛顿法算法 QN 的第二步里执行式(4.25)的精确一维搜索时，条件(4.28)恒成立．其原因是，$x^{(k+1)} = x^{(k)} + t^{(k)}d^{(k)}$ 由式(4.25)确定，故有

$$\nabla f(x^{(k+1)})^T d^{(k)} = 0$$

根据 $s^{(k)} = t^{(k)}d^{(k)}, t^{(k)} > 0, d^{(k)} = -H^{(k)}\nabla f(x^{(k)})$ 及 $H^{(k)}$ 的正定性成立有

$$\begin{aligned}(s^{(k)})^T y^{(k)} &= t^{(k)}(d^{(k)})^T \left(\nabla f(x^{(k+1)}) - \nabla f(x^{(k)})\right)\\ &= -t^{(k)}(d^{(k)})^T \nabla f(x^{(k)})\\ &= -t^{(k)}(d^{(k)})^T \left(H^{(k)}\right)^{-1} d^{(k)} > 0\end{aligned}$$

上面的性质(4.29)保证了搜索方向向量 $d^{(k)}$ 恒为目标函数 f 在点 $x^{(k)}$ 的下降方向，所以拟牛顿法可看成是 4.2 节梯度法的一种．

像这节开始讲的那样，把矩阵 $H^{(k)}$ 取成 $\nabla f^2(x^{(k)})^{-1}$ 之近似是拟牛顿法的基本思想．它的实现要求有下面的**正割条件**(secant condition)，也称为**拟牛顿条件**(quasi-Newton condition)

$$H^{(k+1)}y^{(k)} = s^{(k)} \tag{4.30}$$

实际上，如果 f 为二次函数 $f(x) = \frac{1}{2}x^T Bx + c^T x$，则对任意的 x 成立有 $\nabla^2 f(x) = B$，而且

$$y^{(k)} = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}) = B(x^{(k+1)} - x^{(k)}) = Bs^{(k)}$$

所以

$$\nabla^2 f(x^{(k+1)})^{-1} y^{(k)} = s^{(k)}$$

恒成立．于是，即使对一般的非线性函数 f，要求它的海赛矩阵

$\nabla^2 f(x^{(k+1)})$ 的逆矩阵之近似 $H^{(k+1)}$ 满足正割条件 (4.30) 是很自然的. 另外, DFP 公式或者 BFGS 公式所定的 $H^{(k+1)}$ 满足式 (4.30)

下面的定理表明, 当目标函数为凸二次函数时, 拟牛顿法具有在低于变量个数的迭代次数里找出最优解的显著特征

定理 4.5 (针对二次函数的拟牛顿法的有限收敛性) 目标函数为二次函数 $f(x) = \frac{1}{2}x^T Bx - c^T x$, 其海赛矩阵 B 为 $n \times n$ 正定对称. 这时, 利用 DFP 公式或者 BFGS 公式的拟牛顿法最多在 n 次迭代里找出问题 (4.1) 的 (全局) 最优解 $x^* = B^{-1}c$. 另外, 特别地在第 n 次迭代得到最优解时成立 $H^{(n)} = B^{-1}$.

对一般的非线性函数拟牛顿法的理论上的收敛性还不是太明确. 下面的定理, 是其为数不多的结果中的一个.

定理 4.6 (针对凸函数的拟牛顿法的全局收敛性和收敛率) 假设目标函数 f 的海赛矩阵 $\nabla^2 f(x)$ 一致正定, 也即存在某个常数 $\lambda > 0$ 使得

$$y^T \nabla^2 f(x) y \geq \lambda y^T y \quad (y \in R^n) \tag{4.31}$$

对任意的 $x \in R^n$ 成立 [1]. 这时用 DFP 公式由拟牛顿法生成的点列 $\{x^{(k)}\}$ 收敛于问题 (4.1) 的 (全局) 最优解 x^*. 而且, 如果海赛矩阵 $\nabla^2 f(x)$ 关于 x 李普希兹连续, 则

$$\lim_{k\to\infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0 \tag{4.32}$$

成立

---

1) 在这个假定下, 函数 f 成为有唯一最小点的凸函数.

式 (4.32) 成立时称点列 $\{x^{(k)}\}$ **超线性收敛** (superlinear convergence) 于 x^*, 或者称收敛率为超线性. 虽然超线性收敛是介于线性收敛和二次收敛的概念, 但是事实上可以看成接近二次收敛的非常快收敛.

另外, 上述拟牛顿法的算法里假定了执行精确一维搜索, 但在实际计算中执行 4.2 节所述的"非精确"一维搜索 (参照式 (4.6)) 的情况很多.

到现在为止讲的拟牛顿法是对海赛矩阵的逆矩阵进行逐次逼近. 作为别的方法, 也有通过对海赛矩阵本身进行逼近的矩阵 $B^{(k)}$, 解一次方程组

$$B^{(k)}d = -\nabla f(x^{(k)})$$

来求出搜索方向 $d^{(k)}$ 作为更新海赛矩阵的逼近矩阵 $B^{(k)}$ 的方法, 和 $H^{(k)}$ 的情况一样, 下面的两者具有代表性

Davidon-Fletcher-Powell (DFP) 公式:

$$B^{(k+1)} := B^{(k)} + \left(1 + \frac{(s^{(k)})^T B^{(k)} s^{(k)}}{(y^{(k)})^T s^{(k)}}\right) \frac{y^{(k)}(y^{(k)})^T}{(y^{(k)})^T s^{(k)}} - \left(\frac{y^{(k)}(s^{(k)})^T B^{(k)} + B^{(k)} s^{(k)} (y^{(k)})^T}{(y^{(k)})^T s^{(k)}}\right) \tag{4.33}$$

Broyden-Fletcher-Goldfarb-Shanno (BFGS) 公式

$$B^{(k+1)} := B^{(k)} + \frac{y^{(k)}(y^{(k)})^T}{(y^{(k)})^T s^{(k)}} - \frac{B^{(k)} s^{(k)} (s^{(k)})^T B^{(k)}}{(s^{(k)})^T B^{(k)} s^{(k)}} \tag{4.34}$$

把这些方法称为 DFP 或 BFGS 公式是因为, 对于刚才的由 DFP 公式 (4.26) 所定的 $H^{(k+1)}$ 和由上面的式 (4.33) 所定的 $B^{(k+1)}$, 如果 $H^{(k)} = (B^{(k)})^{-1}$, 则成立 $H^{(k+1)} = (B^{(k+1)})^{-1}$, 以及对于 BFGS 公式 (4.27) 和式 (4.34) 也成立同样的关系. 也即, 式 (4.26) 和式

(4.33) 以及式 (4.27) 和式 (4.34) 都可以看成是等价的．另外，如果 $(s^{(k)})^T y^{(k)} > 0$ 则

$$B^{(k)} \text{ 正定对称} \Longrightarrow B^{(k+1)} \text{ 正定对称}$$

的关系成立，而且正割条件

$$B^{(k+1)} s^{(k)} = y^{(k)}$$

也成立 (参照式 (4.29), (4.30))．更有意思的是，更新 $H^{(k)}$ 的 DFP 公式里作

$$y^{(k)} \to s^{(k)},\ s^{(k)} \to y^{(k)},\ H^{(k)} \to B^{(k)},\ H^{(k+1)} \to B^{(k+1)}$$

的替换就成为更新 $B^{(k)}$ 的 BFGS 公式，对更新 $H^{(k)}$ 的 BFGS 公式进行同样的操作可得到更新 $B^{(k)}$ 的 DFP 公式．也即，可以说 DFP 公式和 BFGS 公式有一种对偶关系．

4.5 共轭梯度法

共轭梯度法 (conjugate gradient method, CG 法) 是与拟牛顿法并列的有代表性的无约束最优化方法．该方法不像拟牛顿法那样每次迭代都更新矩阵，所以特别适应于解大规模问题．

在处理一般的非线性最优化问题之前，考察下面的凸二次函数的最小化问题．

$$\text{目标函数：} \frac{1}{2} x^T B x - c^T x \longrightarrow \text{最小} \tag{4.35}$$

这里，x, c 是 n 维向量，B 是 $n \times n$ 正定对称矩阵．该问题有唯一解 $x^* = B^{-1} c$．

对于两个 n 维向量 x, y，当 $x^T B y = 0$ 时，称 x 和 y 关于矩阵 B 共轭或者 B-共轭 (B-conjugate)．特别是当 B 为单位矩阵

时 B- 共轭向量 x、y 相互正交，因此可以把 B- 共轭性看成是推广向量正交性的概念 (参照图 4.5). 另外，容易验证相互 B 共轭的 $k(\leq n)$ 个非零向量 $d^{(0)}, d^{(1)}, \ldots, d^{(k-1)}$ 是线性独立的. 于是，由向量 $d^{(0)}, d^{(1)}, \ldots, d^{(k-1)}$ 所张成的子空间

$$L^{(k)} = \left\{ x \in R^n \,\middle|\, x = \sum_{i=0}^{k-1} \alpha_i d^{(i)}, \right.$$
$$\left. \alpha_i \in R \ (i = 0, 1, \ldots, k-1) \right\} \tag{4.36}$$

的维数是 k. 特别是 n 个非零 B- 共轭向量所张成的子空间 $L^{(n)}$ 与全空间 R^n 一致.

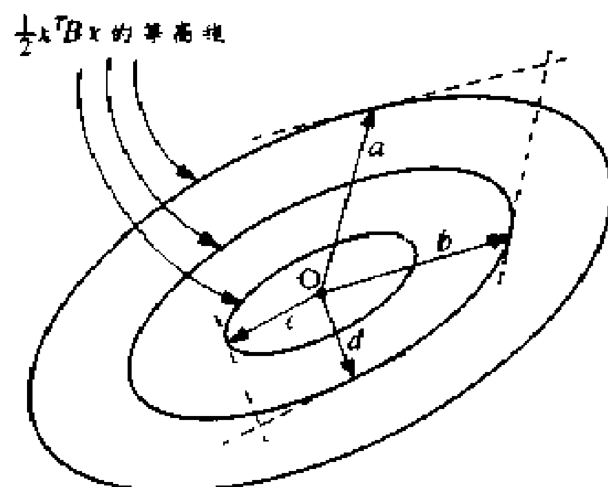


图 4.5 B- 共轭向量的例 (a 和 b B- 共轭，c 和 d B- 共轭)

下面的算法称为**共轭方向法** (conjugate direction methods)，它是后面要讲的共轭梯度法的基础. 另外，问题 (4.35) 的目标函数用 f 表示.

算法 CD (共轭方向法)

第一步 (初始化)：选取适当的初始点 $x^{(0)} \in R^n$，令 $k := 0$.

第二步 (搜索方向的计算): 如果 $k=n$ 则停止. 求出所有的和 $d^{(0)}, \ldots, d^{(k-1)}$ B- 共轭的搜索方向向量 $d^{(k)}$. ($k=0$ 时选取任意的向量为 $d^{(0)}$)

第三步 (一维搜索): 解一维最优化问题

$$\min_{t\in R} f(x^{(k)}+td^{(k)})$$

求出步长 $t^{(k)}$, $x^{(k+1)}:=x^{(k)}+t^{(k)}d^{(k)}$. 令 $k:=k+1$ 回到第二步.

因为目标函数 f 是凸二次函数, 第三步的一维搜索里的步长由

$$t^{(k)}=-\frac{(d^{(k)})^T\nabla f(x^{(k)})}{(d^{(k)})^TBd^{(k)}} \tag{4.37}$$

显式决定. 这里, 和下降法的情况不同, 搜索方向 $d^{(k)}$ 并不限于目标函数的下降方向, 所以步长 $t^{(k)}$ 有可能为负.

对于由共轭方向法生成的点列 $\{x^{(k)}\}$, 成立下面的关系. 点 $x^{(k)}$ 使目标函数在集合 $\hat{L}^{(k)}=x^{(0)}+L^{(k)}$ 上达到最小. 这里 $L^{(k)}$ 是由式 (4.36) 所定义的 R^n 的子空间. 集合 $\hat{L}^{(k)}$ 是平行移动子空间 $L^{(k)}$, 使其原点移至 $x^{(0)}$ 后的集合 (仿射空间). 特别地, 因为 $\hat{L}^{(n)}$ 和全空间 R^n 一致, 所以点 $x^{(n)}$ 是问题 (4.35) 的最优解 (当然, 即使 $k<n$, $x^{(k)}$ 也可能成为最优解). 于是, 下面的定理成立.

定理 4.7 (针对二次函数的共轭方向法的有限收敛性) 如果目标函数 $f(x)=\frac{1}{2}x^TBx-c^Tx$ 的海赛矩阵 B 正定, 则共轭方向法最多在 n 次迭代后找出问题 (4.35) 的最优解 $x^*=B^{-1}c$. 这里 n 是向量 x 的维数.

因为上面所讲的共轭方向法里并没具体指明确定 $B-$ 共轭的搜索方向向量的方法，所以由该方法可以构造不同的算法．实际上，可以证明，把前节考察过的拟牛顿法用于二次函数时生成的搜索方向向量的序列具有 $B-$ 共轭性，所以拟牛顿法成为共轭方向法的特别情况．属于共轭方向法类的另一个有代表性的方法是以下叙述的共轭梯度法．该方法在各次迭代里，利用前次迭代所用的搜索方向及当前点处目标函数的梯度，决定与到目前为止生成的所有搜索方向有 $B-$ 共轭关系的方向向量．

算法 CG (共轭梯度法)

第一步 (初始化)：选取适当的初始点 $x^{(0)} \in R^n$．令 $k := 0$

第二步 (搜索方向的计算)：如果 $\nabla f(x^{(k)}) = 0$ 则停止．如果 $k = 0$ 则令 $d^{(0)} := -\nabla f(x^{(0)})$，如果 $k \geq 1$ 则令

$$d^{(k)} := -\nabla f(x^{(k)}) + \frac{\nabla f(x^{(k)})^T \nabla f(x^{(k)})}{\nabla f(x^{(k-1)})^T \nabla f(x^{(k-1)})} d^{(k-1)}$$

第三步 (一维搜索)：根据式 (4.37) 决定步长 $t^{(k)}$，$x^{(k+1)} := x^{(k)} + t^{(k)} d^{(k)}$．令 $k := k + 1$ 回到第二步．

可以证明算法 CG 生成的搜索方向向量 $d^{(k)}$ $(k = 0, 1, \ldots)$ 相互 $B-$ 共轭，所以由定理 4.7，对某个 $k (\leq n)$ 成立 $x^{(k)} = B^{-1}c$．进一步还知道，在共轭梯度法里到达解所需要的迭代次数一般等于 B 的不同特征值的个数．进行变量的线性变换 $x = Py$，考虑函数

$$\hat{f}(y) = f(Py) = \frac{1}{2} y^T P^T B P y - (P^T c)^T y$$

关于 y 的最小化 (和问题 (4.35) 等价的) 问题．如果选取使矩阵 $P^T B P$ 的特征值分布集中的矩阵 P，则可以期待对函数 $\hat{f}$ 利

用共轭梯度法时在非常少的迭代后得到解. 基于这种思想的方法称为**使用了前处理的共轭梯度法** (preconditioned conjugate gradient methods). 它是加速共轭梯度法的强有力的方法. 作为前处理的矩阵 P 最好使得 P^TBP 接近单位矩阵. 但是为决定这种 P 花费太多的计算是不合适的. 经常用到的选取 P 的方法有. 对 B 的对角成分 b_{ii} 取 $P=\text{diag}(1/\sqrt{b_{ii}})$ 的单纯方法[1] 和对矩阵 B 进行近似 Cholesky 分解计算 $B \simeq L^TL$ (L 是下三角矩阵) 后, 取 $P=(L^T)^{-1}$ 的 **ICCG 法** (incomplete Cholesky conjugate gradient method) 等等.

在理论上共轭梯度法有能在有限次迭代后找出凸二次函数的最小点的性质, 但是实际上, 因为计算误差、$d^{(k)}$ 的 B 共轭性被破坏, 共轭梯度法的有限收敛性不成立的情况很多. 这时, 采取的方法有, 要么迭代即使超过 n 次也照样继续计算, 要么每 n 次迭代后把搜索方向向量 $d^{(k)}$ 修正为 f 的最速下降方向 (称这是**重新启动** (restart)) 继续计算等等.

上面所讲的共轭梯度法算法对一般的非线性函数 f 的最小化也几乎照样适用. 但是, 第二步的一维搜索里, f 如果不是二次函数, 因为步长 $t^{(k)}$ 无法像式 (4.37) 那样用显式表示, 所以有必要用数值的方式解一维最优化问题求出 $t^{(k)}$. 加上这种修正后的共轭梯度法称为 **Fletcher-Reeves 方法**. 可以认为它是非线性最优化的有代表性的方法之一. Fletcher-Reeves 方法是, 由于每 n 次迭代里执行重新启动, f 的局部最优解收敛于 x^*. 进一步若把在重新启动时刻得到的点列记为 $z^{(1)}, z^{(2)}, \ldots$, 则

$$\lim_{j\to\infty} \frac{\|z^{(j+1)} - x^*\|}{\|z^{(j)} - x^*\|} = 0$$

也即证明了相隔 n 次的迭代选到的点列超线性收敛. 但是, 实

1) $\text{diag}(1/\sqrt{b_{ii}})$ 表示以 $1/\sqrt{b_{ii}}$ 为第 i 对角元的对角矩阵.

际上受计算误差的影响，仅能得到类似线性收敛的慢收敛的情况也不少.

于是，从实际的计算效率及稳定性观点来看，共轭梯度法未必比拟牛顿法等要更好. 但是，共轭梯度法中搜索方向的计算里仅仅用到目标函数的梯度，而不必像拟牛顿法那样在每次迭代里更新海赛矩阵的逼近矩阵，并记忆它，还要用到它. 另外一般情况下，虽然问题有大规模**稀疏** (sparse) 结构[1] 倾向，但是即使海赛矩阵是稀疏的，拟牛顿法生成的逼近(逆)矩阵一般都变为非稀疏的，在拟牛顿法里利用问题的稀疏结构是很困难的. 相比之下，当问题的规模大而有稀疏结构时，共轭梯度法有利用该性质极其有效地执行计算的长处.

4.6 带约束最优化问题

该章以下各节考虑下面的带约束最优化问题.

$$
\begin{aligned}
&\text{目标函数} \quad f(x) \to \text{最小} \\
&\text{约束条件} \quad c_i(x) \leq 0 \quad (i = 1,2,\ldots,m)
\end{aligned}
\tag{4.38}
$$

这里，变量 x 是 n 维实向量，目标函数 $f: R^n \to R$ 及约束函数 $c_i: R^n \to R, i = 1,\ldots,m$，都假定为二次连续可微.

这里，如果 f 和 $c_i (i = 1,2,\ldots,m)$ 都是凸函数的话，问题 (4.38) 称为**凸规划问题** (convex programming problem)，可容易证明其局部最优解自动成为全局最优解. 但是在一般的问题里，和无约束情况一样，一般存在有几个局部最优解.

为讨论问题 (4.38) 的最优解的性质，先定义几个术语. 首先，当点 x 是问题 (4.38) 的可行解时，称成立有 $c_i(x) = 0$ 的约

1) 元素几乎都是零的矩阵称为稀疏矩阵

束条件为在点 x 的 **有效约束** (active constraint), 其集合用

$$I(x) = \{ i \mid c_i(x) = 0 \ (i = 1, 2, \ldots, m) \}$$

表示. 进一步, 使有效约束的梯度向量 $\nabla c_i(x)\,(i \in I(x))$ 线性独立的点 x 称为 **正则点** (regular point). 下面的定理所示的条件是特征化非线性最优化问题的最优解的重要条件, 按其发现者命名为 **Kuhn-Tucker 条件** 或者 **Karuxh-Kuhn-Tucker 条件**.

定理 4.8 (Kuhn-Tucker 条件 — 一次必要条件) 如果点 x^* 是问题 (4.38) 的局部最优解而且是正则点, 则存在向量 $u^* = (u_1^*, u_2^*, \ldots, u_m^*)$ 满足

$$\begin{cases} \nabla f(x^*) + \sum\limits_{i=1}^{m} u_i^* \nabla c_i(x^*) = 0 \\ c_i(x^*) \leq 0, \ u_i^* \geq 0, \ u_i^* c_i(x^*) = 0 \ (i = 1, 2, \ldots, m) \end{cases} \tag{4.39}$$

称向量 u^* 为 **拉格朗日乘数** (Lagrange multipliers). 另外, 式 (4.39) 的最后的条件意味着对所有的 i 要么 $u_i^* = 0$, 要么 $c_i(x^*) = 0$, 称之为 **互补条件** (complementarity slackness condition).

Kuhn-Tucker 条件, 在几何学上意味着, 由在局部最优解 x^* 的有效约束的梯度向量 $\nabla c_i(x^*)\,(i \in I(x^*))$ 生成的锥 $\{y \in R^n \mid y = \sum_{i \in I(x^*)} \alpha_i \nabla c_i(x^*), \alpha_i > 0\,(i \in I(x^*))\}$ 包含有向量 $-\nabla f(x^*)$ (图 4.6). Kuhn-Tucker 条件是最优性的必要条件, 一般说来即使成立式 (4.39), 点 x^* 也未必是问题 (4.38) 的局部最优解. 但是, 问题为凸规划问题时, 式 (4.39) 成为最优性的充分必要条件.

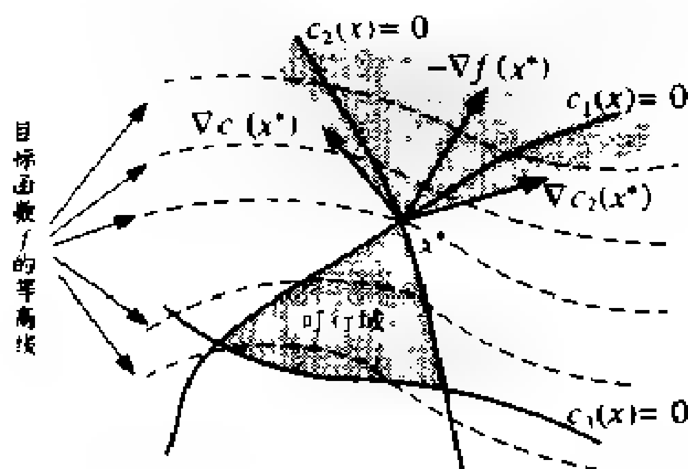


图 4.6 Kuhn-Tucker 条件的几何学解释

最优解 x^* 处有效约束的梯度 $\nabla c_1(x^*)$ 和 $\nabla c_2(x^*)$

生成的锥 (阴影部分) 包含有向量 $-\nabla f(x^*)$

如果考虑函数的二次微分系数(海赛矩阵),可以得到下面的定理 4.9 及定理 4.10 所示的二次最优条件. 这里, $\mathcal{L}: R^{m+n} \to R$ 是

$$\mathcal{L}(x, u) = f(x) + \sum_{i=1}^{m} u_i c_i(x) \tag{4.40}$$

所定义的 **拉格朗日函数** (Lagrangian), 把它的关于 x 的海赛矩阵表示为

$$\nabla_x^2 \mathcal{L}(x, u) = \nabla^2 f(x) + \sum_{i=1}^{m} u_i \nabla^2 c_i(x)$$

另外、定义在(可行)点 x 的可行域的 **切空间** (tangent space) 为

$$M(x) = \left\{ y \in R^n \,\middle|\, \nabla c_i(x)^T y = 0 \, (i \in I(x)) \right\}$$

切空间 $M(x)$ 是对在点 x 的有效约束决定的可行域边界进行线性逼近的集合、它是 R^n 的子空间.

定理 4.9 (二次必要条件) 如果点 x^* 是问题 (4.38) 的局部最优解而且是正则点，则存在满足 Kuhn-Tucker 条件 (4.39) 的拉格朗日乘数 u^*，并且成立

$$y^T \nabla_x^2 \mathcal{L}(x^*, u^*) y \geq 0 \quad (y \in M(x^*))$$

定理 4.10 (二次充分条件) 假定 (x^*, u^*) 满足 Kuhn-Tucker 条件 (4.39)，而且对所有的 $i = 1, 2, \cdots, m$ 成立狭义的互补条件

$$c_i(x^*) = 0 \implies u_i^* > 0$$

这时如果

$$y^T \nabla_x^2 \mathcal{L}(x^*, u^*) y > 0 \quad (y \in M(x^*),\ y \neq 0)$$

则 x^* 是问题 (4.38) 的局部最优解.

定理 4.9 和定理 4.10 是 4.1 节所述的对无约束问题的二次最优条件推广后的结果，但是和无约束情况不同，这里限定在可行域的切空间 $M(x^*)$ 上考虑拉格朗日函数的海赛矩阵之（半）正定性. 对此可说明如下：检查点 x^* 的最优性时，可以考虑在线性逼近可行域的子空间 $M(x^*)$ 上观察目标函数的情况. 因为线性逼近丢失了约束函数的高次信息，为补全它，利用目标函数和各约束函数的加权和的拉格朗日函数的海赛矩阵表现整个问题的二次信息. 也即，在线性逼近约束条件后，所有的非线性成分全综合表示在拉格朗日函数里. 这时拉格朗日乘数 u_i 表示的是问题里各约束条件的权（相对重要度）.

4.7 逐次二次规划法

针对带约束非线性最优化问题，迄今为止提出了各种各样的解法. 这节介绍其中现在被认为是最有效方法之一的逐次二

次规划法 (successive quadratic programming, 也称 **SQP 法**). 所谓**二次规划问题** (quadratic programming problem) 是约束函数为线性，目标函数为二次的特殊非线性规划问题. 特别是，针对凸目标函数的二次规划问题，存在有能在有限次运算里找出最优解的有效算法. 顾名思义，逐次二次规划法是对给定的一般非线性规划问题用二次规划问题进行逐次逼近，生成收敛于待求的最优解的点列的方法.

针对带约束最优化问题 (4.38) 的逐次二次规划法里，第 k 次迭代中给定点 $x^{(k)} \in R^n$ 时，作为子问题解下面的二次规划问题.

$$
\begin{aligned}
&\text{目标函数：}\quad \nabla f(x^{(k)})^T d + \tfrac{1}{2} d^T B^{(k)} d \longrightarrow \text{最小} \\
&\text{约束条件：}\quad c_i(x^{(k)}) + \nabla c_i(x^{(k)})^T d \leq 0 \quad (i = 1, 2, \ldots, m)
\end{aligned}
\tag{4.41}
$$

这里 $B^{(k)}$ 是 $n \times n$ 正定对称矩阵. 如果用 $v \in R^m$ 表示拉格朗日乘数，那么针对该二次规划问题的 Kuhn-Tucker 条件可写成，

$$
\begin{cases}
\nabla f(x^{(k)}) + B^{(k)} d + \displaystyle\sum_{i=1}^{m} v_i \nabla c_i(x^{(k)}) = 0 \\
c_i(x^{(k)}) + \nabla c_i(x^{(k)})^T d \leq 0,\ v_i \geq 0 \quad (i = 1, 2, \ldots, m) \\
v_i \left\{ c_i(x^{(k)}) + \nabla c_i(x^{(k)})^T d \right\} = 0 \quad (i = 1, 2, \ldots, m)
\end{cases}
\tag{4.42}
$$

另外，根据矩阵 $B^{(k)}$ 的正定性，问题 (4.41) 为凸规划问题，只要可行域不是空集合就保证有唯一的最优解. 以下假定问题 (4.41) 的可行域为非空集合，满足 Kuhn-Tucker 条件 (4.42) 的向量对表示为 $(d^{(k)}, v^{(k)})$.

现在，假设解子问题 (4.41) 得到 $(d^{(k)}, v^{(k)})$. 这时，如果 $d^{(k)} = 0$, 则把 $(d, v) = (0, v^{(k)})$ 代入到式 (4.42) 里和式 (4.39) 比较后，可以知道 $(x, u) = (x^{(k)}, v^{(k)})$ 满足针对问题 (4.38) 的 Kuhn-Tucker 条件 (4.39). 也即，因为 $x^{(k)}$ 是满足问题 (4.38) 的最优条

件的点，把待求的解看成是已得到者，结束迭代.

另一方面，如果 $d^{(k)} \neq 0$，把向量 $d^{(k)}$ 作为搜索方向，通过接近问题 (4.38) 最优解来确定下一步的点 $x^{(k+1)}$，与目标函数最小化的同时也必须考虑满足约束条件.（现在的点 $x^{(k)}$ 未必是问题 (4.38) 的可行解.）因此，作为实现这两个不同目标的尺度，引进**罚函数** (penalty function)

$$F_r(x) = f(x) + r\sum_{i=1}^{m} \max\{0, c_i(x)\}$$

这里 $r > 0$ 是称为惩罚参数的常数. 函数 F_r 的第二项，当点 x 是问题 (4.38) 的可行解时为 0，否则取相应于约束条件的不满足度的正值，所以可以认为是约束条件遭到破坏时受到的惩罚. 也就是，借助于最小化罚函数 F_r，同时达到问题 (4.38) 的约束条件的满足化以及目标函数的最小化目标. 于是特别地，当 (x^*, u^*) 满足针对问题 (4.38) 的二次充分条件（定理 4.10）时，选取足够大的惩罚参数，使得 $r > \max\{u_1^*, \cdots, u_m^*\}$，则 x^* 成为无约束最优化问题

$$\text{目标函数：} F_r(x) \longrightarrow \text{最小}$$

的局部最优解. 于是，在设定 r 为足够大值的基础上，通过减少罚函数 F_r 的值来确定下一步的点 $x^{(k+1)}$，就可以期望能够得到所要求的最优解的更好的近似解.

很幸运，可以证明子问题 (4.41) 的解 $d^{(k)}$ 是（当 r 足够大时）函数 F_r 在点 $x^{(k)}$ 的下降方向. 于是，通过对函数 F_r 进行一维搜索，可以确定点 $x^{(k+1)} = x^{(k)} + td^{(k)}$ 使之满足 $f(x^{(k+1)}) < f(x^{(k)})$.

综述以上计算过程，可以得到下面的算法.

算法 SQP (逐次二次规划法)

第一步 (初始化): 选取适当的初始点 $x^{(0)} \in R^n$, $n \times n$ 正定对称矩阵 $B^{(0)}$, 足够大的常数 $r > 0$. 令 $k := 0$

第二步 (计算子问题的解): 解二次规划问题 (4.41), 求出满足式 (4.42) 的向量对 $(d^{(k)}, v^{(k)})$. (如果 $d^{(k)} = 0$ 则结束计算.)

第三步 (一维搜索): 解一维最优化问题

$$\min_{t>0} F_r(x^{(k)} + td^{(k)})$$

求出步长 $t^{(k)}$, $x^{(k+1)} := x^{(k)} + t^{(k)}d^{(k)}$.

第四步 (矩阵 $B^{(k)}$ 的更新): 更新 $B^{(k)}$, 确定新的正定对称矩阵 $B^{(k+1)}$. 令 $k := k + 1$ 回到第二步.

算法的第四步里有关矩阵 $B^{(k)}$ 的更新方法在后面要讲. 在此之前, 先综述有关该算法的全局收敛性的结果.

定理 4.11 (逐次二次规划法的全局收敛性) 假定由逐次二次规划法的算法生成的点列 $\{x^{(k)}\}$, $\{d^{(k)}\}$, $\{v^{(k)}\}$ 有界, 进一步矩阵列 $\{B^{(k)}\}$ 一致有界而且正定, 也即存在满足

$$\lambda d^T d \le d^T B^{(k)} d \le \Lambda d^T d \quad (d \in R^n,\ k = 0, 1, \ldots)$$

的常数 $\Lambda > \lambda > 0$. 这时, 如果选取的惩罚参数 r 充分大, 则 $\{x^{(k)}\}$ 的任意的聚点 x^* 满足针对问题 (4.38) 的 Kuhn-Tucker 条件.

逐次二次规划法的收敛速度在很大程度上依赖于如何确定矩阵 $B^{(k)}$. 下面说明, 针对仅包含下面的等式约束的问题, 基于拟牛顿法的思想来确定矩阵 $B^{(k)}$ 的方法.

$$\begin{array}{ll} \text{目标函数:} & f(x) \to \text{最小} \\ \text{约束条件:} & c(x) = 0 \end{array} \tag{4.43}$$

这里 $c(x)=(c_1(x),c_2(x),\cdots,c_m(x))^T$. 若把这里的函数 $c:R^n\to R^m$ 的 $n\times m$ 雅可比矩阵记为

$$\nabla c(x)=[\nabla c_1(x),\nabla c_2(x),\cdots,\nabla c_m(x)]$$

那么针对问题 (4.43) 的逐次二次规划法里的子问题成为如下的等式约束二次规划问题 [1].

$$\begin{array}{ll}\text{目标函数:} & \nabla f(x^{(k)})^T d+\frac{1}{2}d^T B^{(k)} d \quad \to \text{ 最小} \\ \text{约束条件:} & c(x^{(k)})+\nabla c(x^{(k)})^T d=0\end{array} \tag{4.44}$$

针对该问题的 Kuhn-Tucker 条件可以写为

$$\begin{cases}\nabla f(x^{(k)})+B^{(k)}d+\nabla c(x^{(k)})v=0\\ c(x^{(k)})+\nabla c(x^{(k)})^T d=0\end{cases}$$

所以其最优解 $d^{(k)}$ 和拉格朗日乘数 $v^{(k)}$ 满足

$$\begin{bmatrix}\nabla f(x^{(k)})\\ c(x^{(k)})\end{bmatrix}+\begin{bmatrix}B^{(k)} & \nabla c(x^{(k)})\\ \nabla c(x^{(k)})^T & 0\end{bmatrix}\begin{bmatrix}d^{(k)}\\ v^{(k)}\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix} \tag{4.45}$$

另一方面，针对问题 (4.43) 的 Kuhn-Tucker 条件成为 [2]

$$\begin{cases}\nabla f(x)+\nabla c(x)u=0\\ c(x)=0\end{cases} \tag{4.46}$$

利用式 (4.40) 所定义的拉格朗日函数 $\mathcal{L}$, 这式可以写成

$$\nabla\mathcal{L}(x,u)=\begin{bmatrix}\nabla_x\mathcal{L}(x,u)\\ \nabla_u\mathcal{L}(x,u)\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix} \tag{4.47}$$

---

1) 二次规划子问题的约束条件的形式，根据原来问题的约束条件决定.

2) 和式 (4.39) 不一样，等式约束的情况下没有对拉格朗日乘数的非负条件.

这里，∇_x 和 ∇_u 分别表示关于变量 x 和 u 的微分. 把式 (4.47) 看成是以 (x, u) 为变量的非线性方程，利用牛顿法则得到下面的迭代式.

$$\begin{bmatrix} x^{(k+1)} \\ u^{(k+1)} \end{bmatrix} := \begin{bmatrix} x^{(k)} \\ u^{(k)} \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} \tag{4.48}$$

这里，$(\Delta x, \Delta u)$ 是下面的一次方程的解.

$$\nabla \mathcal{L}(x^{(k)}, u^{(k)}) + \nabla^2 \mathcal{L}(x^{(k)}, u^{(k)}) \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} = 0 \tag{4.49}$$

但是因为

$$\nabla \mathcal{L}(x^{(k)}, u^{(k)}) = \begin{bmatrix} \nabla f(x^{(k)}) + \nabla c(x^{(k)}) u^{(k)} \\ c(x^{(k)}) \end{bmatrix}$$

$$\nabla^2 \mathcal{L}(x^{(k)}, u^{(k)}) = \begin{bmatrix} \nabla_x^2 \mathcal{L}(x^{(k)}, u^{(k)}) & \nabla c(x^{(k)}) \\ \nabla c(x^{(k)})^T & 0 \end{bmatrix}$$

式 (4.48), (4.49) 可以改写如下：

$$\begin{bmatrix} \nabla f(x^{(k)}) \\ c(x^{(k)}) \end{bmatrix} + \begin{bmatrix} \nabla_x^2 \mathcal{L}(x^{(k)}, u^{(k)}) & \nabla c(x^{(k)}) \\ \nabla c(x^{(k)})^T & 0 \end{bmatrix} \begin{bmatrix} x^{(k+1)} - x^{(k)} \\ u^{(k+1)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

把它与式 (4.45) 进行比较，如果 $B^{(k)} = \nabla_x^2 \mathcal{L}(x^{(k)}, u^{(k)})$，则可以知道，针对问题 (4.43) 的 Kuhn-Tucker 条件（非线性方程）的牛顿法的迭代，与解二次规划问题 (4.44) 求出 $(d^{(k)}, v^{(k)})$ 后，令

$$x^{(k+1)} := x^{(k)} + d^{(k)} \tag{4.50}$$

$$u^{(k+1)} := v^{(k)} \tag{4.51}$$

是等价的. 以上讨论还可以进一步推广到含有不等式约束的问题. 于是，逐次二次规划法的各次迭代里，令 $B^{(k)} := \nabla_x^2 \mathcal{L}(x^{(k)}, u^{(k)})$

$^{(k)}$ 的话，可以期待生成的点列像牛顿法那样有

性.

，以二次规划子问题的解 $d^{(k)}$ 为搜索方向，对罚函数

行 维搜索的情况下，为保证全局收敛性有必要要求矩阵 $B^{(k)}$ 为正定的（定理 4 11）然而因为矩阵 $\nabla_x^2\mathcal{L}(x,u)$ 未必正定（甚至在最优解 (x^*,u^*) 处）（参照定理 4.9 和定理 4.10 的二次最优条件），直接令 $B^{(k)} := \nabla_x^2\mathcal{L}(x^{(k)},u^{(k)})$ 是不合适的．于是，为保证算法的全局收敛性，$B^{(k)}$ 要一直保持正定性，同时为得到快收敛希望尽量选取接近 $\nabla_x^2\mathcal{L}(x^{(k)},u^{(k)})$ 者．

因此，仿照针对无约束问题的拟牛顿法，可以考虑通过逼近矩阵 $\nabla_x^2\mathcal{L}(x^{(k)},u^{(k)})$ 来逐步构成 $B^{(k)}$．比如，令 $s^{(k)} := x^{(k+1)} - x^{(k)}, y^{(k)} := \nabla_x\mathcal{L}(x^{(k+1)},u^{(k+1)}) - \nabla_x\mathcal{L}(x^{(k)},u^{(k+1)})$，利用式 (4.34) 的 BFGS 公式

$$B^{(k+1)} := B^{(k)} + \frac{y^{(k)}(y^{(k)})^T}{(y^{(k)})^T s^{(k)}} \quad \frac{B^{(K)}S^{(K)}(S^{(K)})^t b^{(K)}}{(S^{(K)})^t b^{(K)} S^{(K)}} \tag{4.52}$$

更新 $B^{(k)}$ 的话，可以期待 $B^{(k)}$ 渐渐接近于 $\nabla_x^2\mathcal{L}(x^{(k)},u^{(k)})$．但是，根据式 (4.52) 来更新矩阵 $B^{(k)}$ 时，在

$$(y^{(k)})^T s^{(k)} > 0$$

不成立时无法保证 $B^{(k)}$ 的正定性．这个不等式虽然在无约束问题的情况下通常成立（参照 4.4 节），但是，在带约束问题里经常出现不成立的情况．因此，令

$$\theta = \begin{cases} 1 & \text{当}(y^{(k)})^T s^{(k)} \geq 0.2\,(s^{(k)})^T B^{(k)} s^{(k)}\text{时} \\ \dfrac{0.8\,(s^{(k)})^T B^{(k)} s^{(k)}}{(s^{(k)})^T B^{(k)} s^{(k)} - (y^{(k)})^T s^{(k)}} & \text{否则} \end{cases}$$

确定

$$\tilde{y}^{(k)} := \theta y^{(k)} + (1-\theta)s^{(k)}$$

利用把 BFGS 公式 (4.52) 的 $y^{(k)}$ 换成 $\tilde{y}^{(k)}$ 后的式子更新 $B^{(k)}$，就可以恒保持 $B^{(k)}$ 的正定性.

逐次二次规划法像无约束最优化里的拟牛顿法一样具有快速收敛性的前提条件是，对足够大的 k 有 $x^{(k+1)}=x^{(k)}+d^{(k)}$ (参照式 (4.50)). 这意味着在逐次二次规划法的算法里取步长 $t^{(k)}:=1$. 因此，实际上

$$F_r(x^{(k)}+d^{(k)})<F_r(x^{(k)}) \tag{4.53}$$

也即步长为 $t=1$, 罚函数 F_r 的值减少时，经常优先采用步长 $t=1$ 来修正一维搜索.

无约束最优化里的牛顿法和拟牛顿法中，一般当 k 充分大时如果 $x^{(k)}$ 接近解 x^* 的话，即使取步长 $t=1$, 目标函数也减少. 但是在用到不可微罚函数 F_r 的逐次二次规划法中，有可能不论 $x^{(k)}$ 如何接近解 x^*, 式 (4.53) 都不成立. 这种现象称为 **Maratos 效应** (Maratos effect), 它是拟牛顿法的特长也成为损失超线性收敛性的原因. 已经知道，通过修补可以消除 Maratos 效应. 让二次规划子问题的解到二阶为止近似满足原问题的约束条件即可. 但是因为说明起来有些复杂，在此不作更深入的介绍.

4.8 接近点法

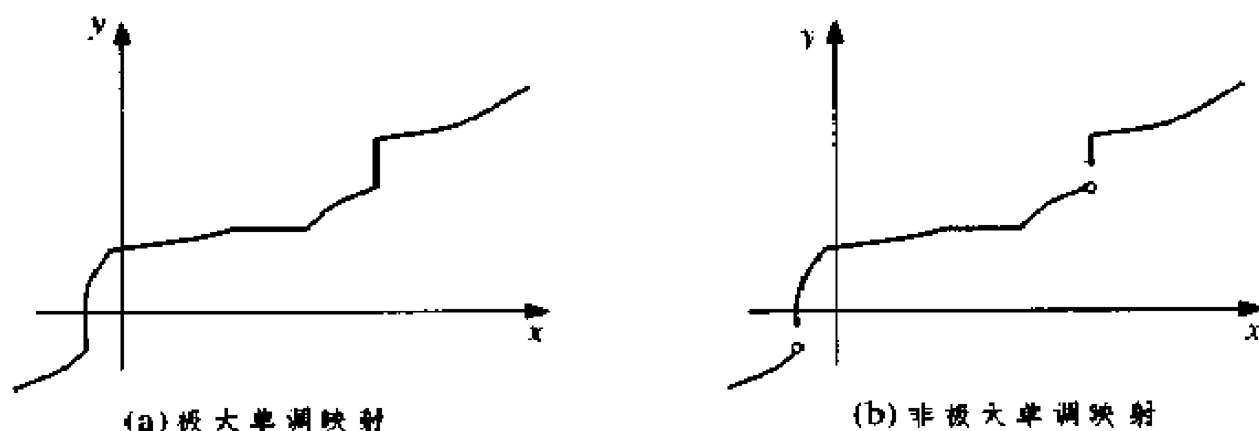
非线性最优化里，一直是古典的解析学起着重要作用，但是近年来，基于**凸分析** (convex analysis) 的各种概念的算法的研究活跃起来. 在数学上它们是非常精练的方法，同时通过灵活利用问题的结构，与针对并行计算的分解算法相关联，这是个很有意义的特征. 这里介绍这类方法之一，**接近点法** (proximal

point algorithm).

考虑把空间 R^n 的各点 x 对应到 R^n 的某个子集合 $T(x)$ 的点–集映射[1] T. 进一步，假定映射 T 满足

$$y \in T(x), y' \in T(x') \Longrightarrow (y - y')^T(x - x') \geq 0. \tag{4.54}$$

满足式 (4.54) 的映射称为**单调** (monotone) 映射. 另外，考虑单调映射 T 的图形 $\mathrm{Graph}(T) = \{(x, y) \in R^n \times R^n \mid y \in T(x)\}$ 时，如果不存在单调映射 T', 使 $\mathrm{Graph}(T) \subset \mathrm{Graph}(T')$ 及 $\mathrm{Graph}(T) \neq \mathrm{Graph}(T')$ 成立，则称 T 为**极大单调** (maximal monotone) 的. 图 4.7 里示意了空间 R^n 的维数 n 为 1 的情况下的单调映射的例子. $n = 1$ 的时候，单调映射的图形成为右上升的，可以看出它对应于通常的单变量实数值函数情况下的单调增加性. 但是，因为现在考虑的是点–集映射，如图所示，图形一般含有垂直线段. 这表示在该 x 处 $T(x)$ 的成分不是一个点. 另外，图 4.7 (b) 的图形是 (a) 的图形的真子集，所以 (b) 的映射是单调的但不是极大的.



(a) 极大单调映射　　(b) 非极大单调映射

图 4.7　单调映射的图例 ($n = 1$ 的情况)

1) 对于某 x 可以有 $T(x) = \emptyset$. 特别是把 $T(x) \neq \emptyset$ 的 x 的集合称为 T 的定义域. 另外，可以认为通常的映射 (点–点映射) 是 $T(x)$ 恒由一个元素构成的特殊情况.

接近点法是求出极大单调的点－集映射 T 的零点、也即满足

$$0 \in T(x) \tag{4.55}$$

的点 $x^* \in R^n$ 的方法. 如果映射 T 是点－点映射的话，式 (4.55) 就是通常的非线性方程，但是像下面要讲的那样，式 (4.55) 的形式所表示的问题包括含有最优化问题在内的非常广泛的问题类. 由此，接近点法也可以说是适用范围极其广泛的方法.

首先，给定实数值凸函数 $f: R^n \to R$ 时，令 $T(x) = \{\nabla f(x)\}$. 请注意 T 满足单调性条件 (4.54). 另外，这时式 (4.55) 可写为 $\nabla f(x) = 0$. 它是针对凸最小化问题

$$\text{目标函数：} f(x) \longrightarrow \text{最小} \tag{4.56}$$

的最优性的充分必要条件. 于是、这种情况下，找出满足式 (4.55) 的 x 的问题和最小化问题 (4.56) 等价. 而且因为 T 是点－集映射，所以即使在函数 f 不可微时候 [1]，对于次微分 $\partial f(x)$，若令 $T(x) = \partial f(x)$，则式 (4.55) 仍然表示最小化问题 (4.56) 的最优条件 $0 \in \partial f(x)$.

进一步，即使对于有闭凸可行域 S 的一般凸最优化问题

$$\begin{aligned} &\text{目标函数：} f(x) \longrightarrow \text{最小} \\ &\text{约束条件：} x \in S(\subseteq R^n) \end{aligned} \tag{4.57}$$

---

1) 如果函数 $f: R^n \to R$ 为凸函数，则在任意点 x 处至少存在一个向量 $y \in R^n$ 满足

$$f(z) - f(x) \geq y^T(z - x) \quad (z \in R^n).$$

称这种 y 为凸函数 f 在点 x 的**次梯度** (subgradient)，称全体次梯度 y 的集合为**次微分** (subdifferential) 记为 $\partial f(x)$. 特别是，如果在点 x 处 f 可微，则 $\partial f(x) = \{\nabla f(x)\}$. 另外，把凸函数的次微分 $\partial f(x)$ 看成是 R^n 到 R^n 的点－集映射的话，则知道它为极大单调映射.

定义可行域 S 在点 x 的**法线锥** (normal cone) 为

$$N_S(x) = \{y \in R^n \mid y^T(z - x) \le 0 \ (z \in S),\}$$

令 $T(x) = \nabla f(x) + N_S(x)$, 则式 (4.55) 可写成

$$-\nabla f(x) \in N_S(x) \tag{4.58}$$

如图 4.8 所示, 式 (4.58) 是针对带约束问题 (4.57) 的最优条件 (参照 4.6 节的图 4.6).

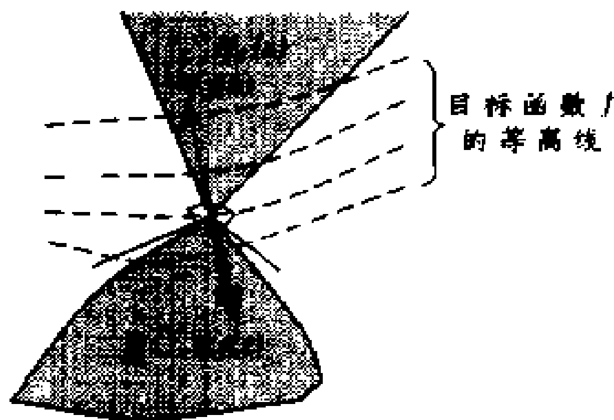


图 4.8 针对问题 (4.45) 的最优条件 (4.46)

作为带约束最优化问题作进一步推广, 有被称为**变分不等式问题** (variational inequality problem) 的问题. 变分不等式问题, 是给定闭凸集合 $S \subset R^n$ 和映射 $F : R^n \to R^n$ 时 [1), 求出满足

$$F(x)^T(z - x) \ge 0 \quad (z \in S) \tag{4.59}$$

的点 $x \in S$ 的问题. 特别是, $S = R^n$ 时, 式 (4.59) 就是非线性方程 $F(x) = 0$, $S = \{x \in R^n \mid x_i \ge 0 \ (i = 1, 2, \ldots, n)\}$ 的时候归为**互补问题** (complementarity problem), 也即求出满足

$$x \ge 0, \ F(x) \ge 0, \ x^T F(x) = 0$$

1) F 也可以是点–集映射. 这个时候问题成为求出对某个 $y \in F(x)$ 有 $y^T(z - x) \ge 0 \ (z \in S)$ 的点 $x \in S$.

的点 x 的问题. 另外, $F(x)$ 等于某凸函数的梯度 $\nabla f(x)$ 时, 变分不等式 (4.59) 可以看成等价于针对函数 f 在可行域 S 上的最小化问题 (4.57) 的最优条件 (4.58). 这样, 变分不等式问题 (4.59) 可以说是非常广泛的问题类, 但是该问题也能通过令 $T(x) = F(x) + N_S(x)$, 归结为问题 (4.55) 的形式.

现在回过头来重新考虑求极大单调的点-集映射 T 的零点 x^* 的问题 (4.55). 针对问题 (4.55) 的接近点法是通过反复迭代

$$x^{(k+1)} \simeq (I + c^{(k)}T)^{-1}(x^{(k)}) \tag{4.60}$$

生成收敛于解 x^* 的点列 $\{x^{(k)}\}$ 的非常单纯的方法. 这里, 式 (4.60) 里用到的 $\{c^{(k)}\}$ 是预先适当确定的正常数的非减少序列. 另外, T 为极大单调时, 对任意的 $y \in R^n$ 和 $c > 0$ 恒存在有唯一的满足

$$y \in (I + cT)(x)$$

的 $x \in R^n$. 也即, 点-集映射 $I + cT$ 的逆映射 $(I + cT)^{-1}$ 是以全空间 R^n 为定义域的点-点映射. 于是, 接近点法的各次迭代里, 式 (4.60) 的右边为唯一确定的向量.

为更详细了解接近点法, 考虑在迭代 (4.60) 里正确求出 $(I + c^{(k)}T)^{-1}(x^{(k)})$ 后令它为 $x^{(k+1)}$ 的情况. 这时

$$x^{(k+1)} = (I + c^{(k)}T)^{-1}(x^{(k)})$$

意味着

$$x^{(k)} \in (I + c^{(k)}T)(x^{(k+1)}) = x^{(k+1)} + c^{(k)}T(x^{(k+1)})$$

也即

$$-\frac{x^{(k+1)} - x^{(k)}}{c^{(k)}} \in T(x^{(k+1)})$$

考虑 $n = 1$ 的情况, 这个关系表示, 通过点 $(x, y) = (x^{(k)}, 0)$ 斜率为 $-1/c^{(k)}$ 的直线 $y = -(x - x^{(k)})/c^{(k)}$ 和映射 T 的图形

$\{(x,y)\mid y\in T(x)\}$ 的交点的 x 坐标是 $x^{(k+1)}$ (进一步知道，映射 T 为极大单调的话，这种交点恒为一点，也即，上面所讲的 $(I+cT)^{-1}$ 成为点—点映射．) 图 4.9 所示的是针对 $n=1$ 的问题，$c^{(k)}$ 取定值时由接近点法生成的点列 $\{x^{(k)}\}$ 的情形．

在把接近点法特别运用到最小化问题 (4.57) 的情况下，迭代 (4.60) 意味着近似解最小化问题

$$\begin{aligned}&\text{目标函数：} f(x)+\frac{1}{2c^{(k)}}\|x-x^{(k)}\|^2 \quad\to\quad \text{最小} \qquad (4.61)\\&\text{约束条件：} x\in S(\subseteq R^n)\end{aligned}$$

而运用到变分不等式问题 (4.59) 里就对应于近似解变分不等式问题

$$\left(F(x)+\frac{1}{c^{(k)}}(x-x^{(k)})\right)^T(z-x)\geq 0 \quad (z\in S) \qquad (4.62)$$

问题 (4.61) 和 (4.62) 是与原问题同一类型的问题，但是附加的项有令人满意的性质，在很多情况下易求解[1]，所以可以认为反复近似解这些问题实际上很有意义．

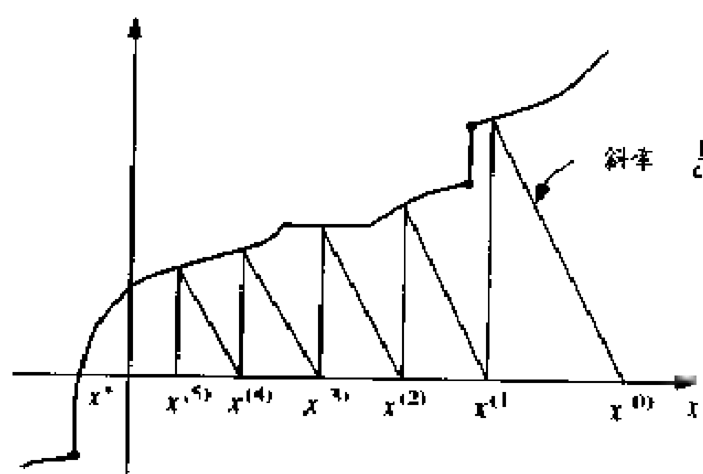


图 4.9　1 维问题的接近点法

(参数 $c^{(k)}$ 取定值 $c>0$ 的情形)

1) 用这种方法来改善问题的性质称为正则化 (regularization)

接近点法的迭代 (4.60) 里，作为 $(I+c^{(k)}T)^{-1}(x^{(k)})$ 之近似求 $x^{(k+1)}$ 时，如果让它满足条件

$$\|x^{(k+1)}-(I+c^{(k)}T)^{-1}(x^{(k)})\|\le\varepsilon_k\quad(k=0,1,\cdots)\tag{4.63}$$

(这里 $\{\varepsilon_k\}$ 是满足 $\sum_{k=1}^{\infty}\varepsilon_k<\infty$ 的正数列)，或者

$$\|x^{(k+1)}-(I+c^{(k)}T)^{-1}(x^{(k)})\|\le\delta_k\|x^{(k+1)}-x^{(k)}\|\quad(k=0,1,\cdots)\tag{4.64}$$

(这里 $\{\delta_k\}$ 是满足 $\sum_{k=1}^{\infty}\delta_k<\infty$ 的正数列)，则可以证明对任意的初始点、生成的点列 $\{x^{(k)}\}$ 收敛于问题 (4.55) 的解 x^*. 或者一般地，迭代 (4.60) 里，$c^{(k)}$ 充分小时，可以认为待求的 $x^{(k+1)}$ 就在当前点 $x^{(k)}$ 的附近. 为直观理解这个事实，只要考虑一下对作为问题 (4.55) 的特殊情况的最小化问题 (4.57)，利用接近点法即可. 这时，$x^{(k+1)}$ 可以通过近似解最小化问题 (4.61) 得到，但是当 $c^{(k)}$ 很小时，认为问题 (4.61) 的解存在于 $x^{(k)}$ 附近是很自然的. (图 4.9 表明，当 $c^{(k)}$ 很小时，$x^{(k)}$ 和 $x^{(k+1)}$ 很接近，当 $c^{(k)}$ 很大时 $x^{(k)}$ 和 $x^{(k+1)}$ 的距离变大.) 于是，把 $x^{(k)}$ 作为初始点用适当的迭代法解问题 (4.61)，可以期待能比较容易地得到近似解 $x^{(k+1)}$. 然而，也可以认为，$x^{(k)}$ 和 $x^{(k+1)}$ 的距离很小时，为收敛于原问题 (4.55) 的解 x^* 需要的接近点法的迭代次数变大. 于是，接近点法的初始迭代里选取小的 $c^{(k)}$，随着迭代进行下去逐渐加大 $c^{(k)}$. 通过这种方法加速收敛到解 x^* 实际上是有效的. 特别地，可以证明，当 $c^{(k)}\to\infty$，在适当的条件下，$\{x^{(k)}\}$ 的收敛速度为超线性. 以上结果可综述如下.

定理 4.12 (接近点法的收敛性) 接近点法 (4.60) 里，假设 $\{c^{(k)}\}$ 为非减少的正数列，$x^{(k+1)}$ 是按条件 (4.63) 或 (4.64) 所

确定的．这时，对任意的初始点 $x^{(0)}$，生成的点列 $\{x^{(k)}\}$ 收敛于问题 (4.55) 的解 x^*．进一步，如果问题 (4.55) 有唯一解 x^*，而且存在对于充分小的任意向量 $w \in R^n$ 和任意的 $x \in T^{-1}(w)$ 成立有满足 $\|x - x^*\| \le \alpha\|w\|$ 的常数 $\alpha > 0$，则利用条件 (4.64) 由接近点法生成的点列 $\{x^{(k)}\}$ 超线性收敛于 x^*

如上所述，接近点法是在非常一般的框架下构成的方法，它与迄今为止开发出的众多方法紧密相连．特别是，虽然在本书省略了讲解但是在非线性规划里有名的**乘数法** (multiplier method, augmented Lagrangian method) 也可看成是针对给定问题的对偶问题所利用的接近点法．

4.9 文献及其它话题

有关非线性最优化方法出版了为数众多的出色的讲解书，这里仅列举 Bertsekas [B82] Dennis, Schnabel [DS83], Fletcher [f87], Gill, Murray, Wright [GMW81] 以及今野，山下 [KY78] 的教科书和 Dennis, Schnabel [DS89], Gill, Murray, Saunders, Wright [GMSW89] 的综述论文．另外，关于非线性最优化里最优条件和对偶定理等的基础理论，除上述文献外还有福岛 [F80], Rockafellar [R70] 等书．进一步，ASNOP [A91], 茨木，福岛 [IF91] 里还给出了各种各样的非线性最优化算法的具体程序．

针对非线性最优化问题迄今为止提出了各式各样的方法，本书受页数的限制不得不限于介绍有代表性者．特别是，关于无约束最优化的各定理，仅有叙述而无详细讲解，这些定理的证明，请参照今野，山下 [KY78] 等等．另外，针对带约束最优化仅涉及了逐次二次规划法，除此之外还有，把带约束问题变

换为无约束问题(的序列)来解的**惩罚法** (penalty method) 和**乘数法** (multiplier method), 把线性规划的单纯形法推广到非线性规划后的**推广既约梯度法** (generalized reduced gradient method, GRG 法) 等等强有力的方法. 有关其详细内容请参照上述教科书. 关于逐次二次规划法的理论上的各性质在 Bertsekas [B82] 和 Fletcher [F87] 里有详细讲解. 另外, 消除逐次二次规划法里的 Maratos 效果的方法在茨木, 福岛 [IF91] 里也出现了.

4.8 节介绍的接近点法是由 Rockafellar [R76] 在非常一般的框架下做的详细研究. 以接近点法为代表的基于凸分析的方法, 在数学上精练, 同时, 它的适用范围很广, 所以最近的研究非常活跃. 特别是, 作为利用了问题结构的并行算法的基础方法, 它非常有效. Bertsekas, Tsitsiklis [BT89] 讲解了包含这种方法在内的各种并行最优化算法. 进一步, 4.8 节简单涉及的变分不等式问题也是最近引人注目的领域, Harker, Pang [HP90] 的综述论文对了解其理论和应用概要很有用. 关于凸分析的诸概念除上述 [F80], [KY78], [R70] 外, 在布川, 中山, 谷野 [FNT91] 里也有讲解.

第 5 章 线性规划：椭球法和内点法

本章介绍针对线性规划问题的多项式时间算法 — 椭球法和内点法. 首先叙述椭球法，它最先在理论上明确线性规划问题恒在多项式时间内可解，因而意义深远. 其次讲解现在作为替代单纯形法的有效方法而引人注目的内点法.

5.1 椭球法

像 1.4 节所述的那样，经验上单纯形法在 $O(m)$ 次的迭代后结束计算. 但是在最坏的情况下计算量成为变量数目的指数函数，因而在理论上不是多项式时间算法. 对此，1979 年俄国数学家 L. G. Khachiyan (Л. Г. Хачиян)[1] 基于和单纯形法完全不同的思想提出称为**椭球法** (ellipsoid method) 的新算法，并证明了该方法在最坏计算量的意义下为多项式时间算法. 然而，虽然椭球法在理论上有胜过单纯形法的性质，但是已经证实了它在实际计算效率方面明显劣于单纯形法，所以还没有达到替代单纯形法的实用化地步. 因此，椭球法的贡献完全停留在理论范围内，但是给线性规划领域带来的影响非常重大. 所以，本节想讲解有关椭球法的基本想法及其理论性质.

椭球法，不是直接处理线性规划问题，而是先判定下面的

1) Хачиян 发音很难用日本语正确标记，很多时候写成カチヤン或者ハチヤン.

联立一次不等式有没有解，如果有解，再用迭代法实际上把它求出来．

$$a_i^T x \le b_i \quad (i = 1, 2, \ldots, m) \tag{5.1}$$

这里，变量 x 是 n 维实向量，为简单起见，假定常数向量 a_i 的各分量及 b_i 为整数．本节后半部分指明求联立一次不等式的解如何与解线性规划问题相关联，在此之前先叙述针对不等式 (5.1) 的椭球法的基本思想．

第 k 次迭代开始时，假定给定了具有如下性质的椭球[1] $E^{(k)} \subset R^n$.

$$\begin{aligned} \bar{X} = \{x \in R^n \mid a_i^T x \le b_i \ (i = 1, 2, \ldots, m)\} \ne \emptyset \\ \Longrightarrow \bar{X} \cap E^{(k)} \ne \emptyset \end{aligned} \tag{5.2}$$

一般地，事先并不知道不等式 (5.1) 有没有解，这里先假定不等式 (5.1) 的解集合 $\bar{X}$ 非空来继续我们的话题．

那么现在把注意力集中到椭球 $E^{(k)}$ 的球心 $x^{(k)}$．如果 $x^{(k)}$ 满足不等式 (5.1)，则已经得出待求的解，从而迭代到此结束．否则任意选取一个满足

$$a_i^T x^{(k)} > b_i \tag{5.3}$$

的 i．图 5.1 针对变量 x 的维数为 2 的时候示意了这么一种状况，直线 $a_i^T x = b_i$ 的左下部分对应于 $a_i^T x \ge b_i$ 的领域，右上部分对应于 $a_i^T x \le b_i$ 领域．于是，根据性质 (5.2) 和式 (5.3)，式 (5.1) 的解应该存在于直线 $a_i^T x = b_i$ 的右上领域和椭球 $E^{(k)}$ 的共同部分 (图的斜线部分) 里．因此，考虑把直线 $a_i^T x = b_i$ 作平行移动使其通过椭球 $E^{(k)}$ 的球心 $x^{(k)}$，成为 $a_i^T x = a_i^T x^{(k)}$,

---

1) n 维空间 R^n 里 $E = \left\{x \in R^n \,\middle|\, (x - \hat{x})^T B (x - \hat{x}) \le 1\right\}$ 表示的集合称为椭球．这里，$\hat{x}$ 是椭球的球心，B 是 $n \times n$ 正定对称矩阵．特别是 $B = \alpha I$ (α 为正数，I 为单位矩阵) 的时候，椭球成为球心为 $\hat{x}$，半径为 $1/\sqrt{\alpha}$ 的球．

则式 (5.1) 的解存在于该直线二分椭球 $E^{(k)}$ 后得到的半椭球 $\hat{E}^{(k)} := E^{(k)} \cap \{x \in R^n \mid a_i^T x \leq a_i^T x^{(k)}\}$ 之中. 于是, 如果进一步考虑包含 $\hat{E}^{(k)}$ 的最小椭球, 则可以保证它一定包含式 (5.1) 的解. 另外, 如果根据

$$x^{(k+1)} := x^{(k)} - \frac{1}{n+1} \frac{B^{(k)} a_i}{\sqrt{a_i^T B^{(k)} a_i}} \tag{5.4}$$

$$B^{(k+1)} := B^{(k)} \cdot \frac{n^2}{n^2 - 1} \cdot \left(B^{(k)} - \frac{2}{n+1} \frac{(B^{(k)} a_i)(B^{(k)} a_i)^T}{a_i^T B^{(k)} a_i} \right) \tag{5.5}$$

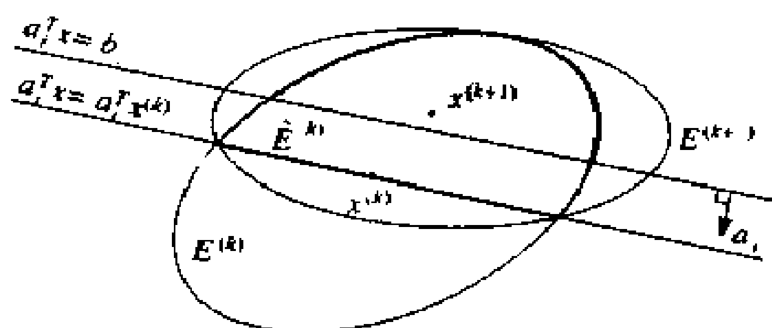


图 5.1 椭球法的迭代

确定 $x^{(k+1)}$ 和 $B^{(k+1)}$ 的话, 包含半椭球 $\hat{E}^{(k)}$ 的最小椭球由

$$E^{(k+1)} = \left\{x \in R^n \,\middle|\, (x - x^{(k+1)})^T B^{(k+1)} (x - x^{(k+1)}) \leq 1 \right\}$$

给出, 而且, 如果椭球 E 的体积用 $\mathrm{vol}(E)$ 表示的话, 则成立关系

$$\frac{\mathrm{vol}(E^{(k+1)})}{\mathrm{vol}(E^{(k)})} = \frac{n}{n+1} \cdot \left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}} < 2^{-\frac{1}{2(n+1)}} \tag{5.6}$$

于是, 若重复上面的手续, 在保持性质 (5.2) 的同时, 逐渐生成体积按低于恒定比例 $2^{-1/2(n+1)}$ (<1) 之速度减少的椭球序列,

从而使得包含不等式 (5.2) 的解的范围逐步得到限定[1].

以上是椭球法的概略，其计算顺序可以总结如下.

算法 ELLIPSOID (椭球法)

第一步 (初始化): 选取具有性质 (5.2) 的足够大的椭球 $E^{(0)}$ 确定最大迭代次数 K. 令 $k := 0$.

第二步 (解的判定): 如果椭球 $E^{(k)}$ 的球心 $x^{(k)}$ 满足不等式 (5.1) 则结束计算.

第三步 (椭球的更新): 选取一个满足 $a_i^T x^{(k)} > b_i$ 的 i. 根据式 (5.4), (5.5), 求出新的椭球 $E^{(k+1)}$, 令 $k := k+1$. 如果 $k = K$ 则结束计算，否则回到第二步.

执行该算法时，在初始化阶段，有必要确定具有性质 (5.2) 的椭球 $E^{(0)}$ 和最大迭代次数 K. 因此，作为表示不等式 (5.1) 的数据大小的指标，定义[2]

$$L = \sum_{i,j} \lceil \log(|a_{ij}| + 1) + 1 \rceil + \sum_{i} \lceil \log(|b_i| + 1) + 1 \rceil \tag{5.7}$$

它是不等式 (5.1) 的输入数据长度，表示为记忆作为二进制数的所有系数 a_{ij}, b_i 必要存储领域之大小 (位数). 这时，把以原点为球心，2^L 为半径的球记为 $\hat{S}$, 则不等式 (5.1) 有解时，可以

1) 各迭代里，不考虑包含半椭球 $E^{(k)}$ 的椭球，而考虑包含集合 $\{x \in E^{(k)} \mid a_i^T x \le b_i\}$ (图 5.1 的斜线部分) 的最小椭球，同样的讨论在 $E^{(k+1)}$ 上也成立. 显然，这种方式的体积 $\mathrm{vol}(E^{(k)})$ 以更快的速度减少下去，但是各次迭代里椭球的更新公式比起式 (5.4), (5.5) 还要变得复杂一些.

2) 式 (5.7) 里，对数的底是 2, $\lceil a \rceil$ 表示不小于 a 的最小整数.

断言这解集合 $\bar{X}$ 和球 $\hat{S}$ 的共同部分非空.（这个性质可以由不等式系数的整数性导出.）也即，球 $\hat{S}$ 具有性质 (5.2)，所以可以作为初始椭球 $E^{(0)} = \hat{S}$.

其次考虑，最大迭代次数 K 要设定为多大才行. 首先，假定不等式 (5.1) 的解集合和初始椭球的共同部分 $\bar{X} \cap E^{(0)}$ 具有正的体积，其值已知. 这时，如果考虑到根据椭球法的性质恒成立有

$$\bar{X} \cap E^{(0)} \subset E^{(k)}$$

则式 (5.1) 的解应该在椭球 $E^{(k)}$ 的体积变得比集合 $X \cap E^{(0)}$ 的体积更小之前一定找出. 由于式 (5.6)，椭球 $E^{(k)}$ 的体积至少以恒定比例 $2^{-1/2(n+1)} (< 1)$ 减少下去，根据（已知的）初始椭球和集合 $X \cap E^{(0)}$ 的体积，可以估计出椭球 $E^{(k)}$ 的体积变得比集合 $\bar{X} \cap E^{(0)}$ 的体积更小所需要的迭代次数. 因此把它作为最大迭代次数 K 即可.

但是一般说来，不等式 (5.1) 并不一定有解，即使有解，解集合 $\bar{X}$ 的体积也可能为零.（请想像一下 X 为 3 维空间内的无"厚度"的平面集合，或者线段以及只有一点的集合的情形.）这种时候不能直接利用上面有关估计最大迭代次数的讨论，但是通过利用下述技巧，问题可以得到解决.

针对不等式 (5.1)，定义下面的不等式.

$$2^L a_i^T x \leq 2^L b_i + 1 \quad (i = 1, 2, \ldots, m) \tag{5.8}$$

考虑这个不等式两边除以 2^L 后的等价不等式，它是不等式 (5.1) 的右边放大 2^{-L} 后的结果. 但是，因为 2^{-L} 的值充分小，故仅在不等式 (5.1) 有解时不等式 (5.8) 才有解. 这是个很重要的关系.（它是基于不等式的系数全部为整数的假定得到的.）如果得到了不等式 (5.8) 的解，则可以利用它计算出不等式 (5.1)

的一个解．根据这些事实，不等式 (5.1) 和不等式 (5.8) 可以看成"等价"，所以下面考虑对不等式 (5.8)，而不是对不等式 (5.1)，利用椭球法．

如果不等式 (5.1) 有解 $\bar{x}$，则可以证明对于 $\|x-\bar{x}\| \le 2^{-2L}$ 的任意 x 都成为不等式 (5.8) 的解，所以知道不等式 (5.8) 的解集合包含有以点 $\bar{x}$ 为球心以 2^{-2L} 为半径的球．也即，不等式 (5.1) 有解时，可保证不等式 (5.8) 的解集合至少有半径为 2^{-2L} 的球的体积．另外，n 维空间 R^n 里，把单位球（半径为 1 的球）的体积记为 $\hat{v}$，则半径为 r 的球的体积成为 $r^n\hat{v}$．这样一来，如果初始椭球 $E^{(0)}$ 是半径为 2^L 的球 S，则因为椭球 $E^{(k)}$ 的体积在每次迭代里以 $2^{-1/2(n+1)}$ 的比例减少下去，所以使它第一次变得比半径为 2^{-2L} 的球的体积小所需要的迭代次数上限由满足下式的最小 k 给出．

$$\left(2^{-\frac{1}{2(n+1)}}\right)^k \left(2^L\right)^n \hat{v} < \left(2^{-2L}\right)^n \hat{v} \tag{5.9}$$

满足不等式 (5.9) 的最小 k 是 $6n(n+1)L$，可以把它作为最大迭代次数 K．概括以上讨论，得到下面的定理．

定理 5.1（椭球法的多项式性） 针对不等式 (5.8) 利用椭球法时，如果原来的不等式 (5.1) 有解，则一定在 $K=6n(n+1)L$ 以下次数的迭代内，椭球的球心满足式 (5.8) 从而结束计算．反之，如果进行 K 次迭代后仍然结束不了计算，则不等式 (5.1) 无解．最大迭代次数 K 是问题规模的多项式阶 $O(n^2L)$，每次迭代的计算量也显然可以控制在问题规模的多项式内，所以椭球法是多项式时间算法．

椭球法的计算量依赖于问题的数据长度 L，故不是多项式算法．

到现在为止是把椭球法当做不等式 (5.1) 的解法进行考虑的，但是稍微下点工夫后，也可以用它来解线性规划问题．这样的方法已经知道有好几个．其中有一个是利用 1.7 节所说的线性规划问题的原问题和对偶问题，把问题变换成为不等式 (5.1) 的形式．现在，假设给定了要解的线性规划问题为

$$\begin{aligned}&\text{目标函数：}\quad c^Tx \longrightarrow \text{最小}\\&\text{约束条件：}\quad Ax \geq b,\ x \geq 0\end{aligned}\tag{5.10}$$

该问题的对偶问题是

$$\begin{aligned}&\text{目标函数：}\quad b^Tw \longrightarrow \text{最大}\\&\text{约束条件：}\quad A^Tw \leq c,\ w \geq 0\end{aligned}\tag{5.11}$$

根据 1.7 节的定理 1.4 (弱对偶定理) (i)，对于问题 (5.10) 和问题 (5.11) 的任意可行解 x, w，着眼于恒成立的不等式 $c^Tx > b^Tw$．考虑包含其反向不等式的如下联立不等式组．

$$\begin{cases}Ax > b, & x \geq 0\\A^Tw \leq c, & w \geq 0\\c^Tx \leq b^Tw\end{cases}\tag{5.12}$$

如果该不等式存在解 x, w，则根据上面所述的事实，必然成立 $c^Tx = b^Tw$，由定理 1.4 的 (iv)，x, w 分别是问题 (5.10) 和问题 (5.11) 的最优解．这样一来，如果可能用椭球法求出不等式 (5.12) 的解，则可以 (和对偶问题 (5.11) 最优解同时) 得到问题 (5.10) 的最优解．另外，虽然不等式 (5.12) 的规模比起问题 (5.10) 要大，但是最多不会超过常数倍．于是，在对不等式 (5.12) 利用椭球法的情况下，计算量上的评价成为关于原线性规划问题 (5.10) 的规模的多项式．

5.2 Karmarkar 法

N. Karmarkar 在 1984 年，提出了针对线性规划问题的新的多项式时间算法，并宣告它在实际计算效率方面也比单纯形法好. 其后，受 Karmarkar 方法启发，所谓**内点法** (interior point method 或者 interior method) 范畴里的各式各样的方法被提出来了. 单纯形法是一边寻找可行域 (凸多面体) 的顶点一边生成到达最优解的点列 (可行基解的序列). 相比之下，内点法像其名字一样，是经过可行域内部生成收敛于最优解的点列的方法. 另外，内点法算法中有很多被指出与非线性最优化里有名的投影梯度法，牛顿法以及罚函数法等有着密切的关系. 这节叙述 **Karmarkar 法** (Karmarkar's method) 的思想 [1]，下节以后，说明有代表性的内点法算法——仿射变换法及其多项式时间版本.

Karmarkar 法处理如下形式的线性规划问题.

$$
\begin{aligned}
&\text{目标函数:} \quad c^T x \longrightarrow \text{最小} \\
&\text{约束条件:} \quad Ax = 0 \\
&\qquad\qquad\quad\ \ e^T x = 1, \ x > 0
\end{aligned}
\tag{5.13}
$$

这里，变量 x 是 n 维向量，A 是 $m \times n$ 常数矩阵，c 是 n 维常数向量，e 是所有分量为 1 的 n 维向量 $e = (1, 1, \ldots, 1)^T$，A 和 c 的分量全部为整数.

问题 (5.13) 的可行域可以表示为如下两个集合的共同部分

$$
\Omega = \{x \in R^n \mid Ax = 0\} \tag{5.14}
$$

$$
S = \{x \in R^n \mid e^T x = 1, \ x \geq 0\} \tag{5.15}
$$

1) 本节所说的 Karmarkar 法指的是 1984 年 Karmarkar 最初提出的方法.

这里，集合 Ω 是空间 R^n 的子空间，S 是称为**单纯形** (simplex) 的 $(n-1)$ 维特殊形状的集合[1]. 特别地，满足 $x>0$ 的可行解 $x \in \Omega \cap S$ 称为问题 (5.13) 的**可行内点** (feasible interior point).

进一步，对问题 (5.13) 做如下假定.

1. $Ae=0$ 成立. 也即, 单纯形 S 的重心 $e/n=(1/n, 1/n, \ldots, 1/n)^T$ 是问题 (5.13) 的可行解.

2. 问题 (5.13) 的最小值为 0.

以第 1 章处理标准形问题的习惯眼光来看，问题 (5.13) 的形状非常特别，针对它的假定也留有不自然的印象. 但是即使按这种形式设定问题，也不会失去理论上的一般性[2].

像上面所讲的那样，Karmarkar 法是生成收敛于问题 (5.13) 的最优解的可行内点序列 $\{x^{(k)}\}$ 的方法. 以下说明从给定的可行内点 $x^{(k)}$ 确定下一步的可行内点 $x^{(k+1)}$ 的方法. 首先，对点 $x^{(k)}$，定义把单纯形 S 的点 $x=(x_1, x_2, \ldots, x_n)^T$ 映射到单纯形 S 的点 $y=(y_1, y_2, \ldots, y_n)^T$ 的映射

$$y_j=\frac{x_j/x_j^{(k)}}{\sum_{h=1}^n (x_h/x_h^{(k)})} \quad (j=1, 2, \ldots, n) \tag{5.16}$$

这里，把以 $x^{(k)}$ 的各要素为对角分量的对角矩阵记为

---

1) $n=2$ 的时候 S 是连接两点 (1,0),(0,1) 的线段，$n=3$ 的时候是以三点 (1,0,0),(0,1,0),(0,0,1) 为顶点的正三角形，$n=4$ 的时候是以四点 (1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1) 为顶点的正四面体. 也即，S 的维数比所考虑的空间的维数仅仅小 1

2) 有关把一般问题变换成满足假定 1, 2 的问题形式 (5.13) 的方法请参照 Karmarkar 的论文或者章末给出的参考文献. 但是，这些方法是非常人为的，另外进行这种变换来解问题实际上并不一定有益

$$D^{(k)} = \begin{bmatrix} x_1^{(k)} & 0 & \cdots & 0 \\ 0 & x_2^{(k)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n^{(k)} \end{bmatrix} \tag{5.17}$$

则式 (5.16) 可以用如下向量表示.

$$y = \frac{(D^{(k)})^{-1}x}{e^T(D^{(k)})^{-1}x} \tag{5.18}$$

以下把这个映射表示为 $y = T^{(k)}(x)$. 容易理解，因为映射 $T^{(k)}$ 的逆映射 $(T^{(k)})^{-1}$ 由

$$(T^{(k)})^{-1}(y) = \frac{D^{(k)}y}{e^T D^{(k)}y} \tag{5.19}$$

给出，$T^{(k)}$ 成为单纯形 S 到单纯形 S 上的 1-1 映射. 特别是，通过映射 $T^{(k)}$，单纯形 S 的各顶点映射至同样的顶点，各边映射至同样的边. 另外，单纯形 S 内的任意线段仍然映射为单纯形 S 内的线段，把具有这种性质的映射 (变换) 一般称为**投影变换** (projective transformation). 上面定义的映射 $T^{(k)}$ 是把现在的点 $x^{(k)}$ 映射至单纯形 S 的重心 $e/n = (1/n, \cdots, 1/n)^T$ 的投影变换，成为 Karmarkar 法的重要工具之一 (参照图 5.2).

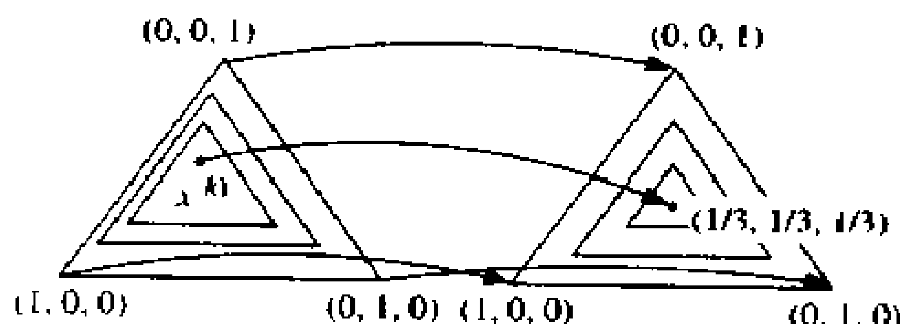


图 5.2 投影变换 $T^{(k)}$ 之例 $(n = 3)$

如果利用投影变换 $T^{(k)}$, 则根据式 (5.19), 问题 (5.13) 的目标函数成为

$$c^T x = \frac{c^T D^{(k)} y}{e^T D^{(k)} y} \tag{5.20}$$

约束条件 $Ax = 0$ 成为

$$Ax = \frac{AD^{(k)} y}{e^T D^{(k)} y} = 0 \tag{5.21}$$

进一步, 注意到 $T^{(k)}$ 把单纯形 S 映射为自身, 对 S 所包含的任意的 y 成立 $e^T D^{(k)} y > 0$, 则可以知道问题 (5.13) 等价于下面的问题.

$$\begin{aligned} &\text{目标函数:} \quad \frac{(c^{(k)})^T y}{e^T D^{(k)} y} \longrightarrow \text{最小} \\ &\text{约束条件} \quad A^{(k)} y = 0 \\ &\qquad\qquad\quad\; e^T y = 1, \; y \geq 0 \end{aligned} \tag{5.22}$$

这里, $c^{(k)}, A^{(k)}$ 分别是

$$c^{(k)} = D^{(k)} c \tag{5.23}$$

$$A^{(k)} = AD^{(k)} \tag{5.24}$$

所定义的 n 维向量, $m \times n$ 矩阵. 另外, 如上所述, 变换 $T^{(k)}$ 把点 $x^{(k)}$ 映射至单纯形 S 的重心 e/n, 根据矩阵 $D^{(k)}$ 的定义以及 $x^{(k)}$ 是问题 (5.13) 的可行解, 成立有

$$A^{(k)} e = AD^{(k)} e = Ax^{(k)} = 0$$

这样一来, 点 e/n 是问题 (5.22) 的可行解.

因此, 问题 (5.22) 里, 考虑从点 e/n 出发, 在满足约束条件的同时, 沿目标函数减少的方向移动. 这里, 定义 $(m+1) \times n$

矩阵 $B^{(k)}$ 为

$$B^{(k)} = \begin{bmatrix} A^{(k)} \\ e^T \end{bmatrix} \tag{5.25}$$

令 $b = (0, \ldots, 0, 1)^T \in R^{m+1}$, 则问题 (5.22) 的等式约束条件可以一起表示为

$$B^{(k)}y = b \tag{5.26}$$

以下为简化讨论，假定

$$\operatorname{rank} B^{(k)} = m + 1 \tag{5.27}$$

这时，$(m+1) \times (m+1)$ 矩阵 $B^{(k)}(B^{(k)})^T$ 是非奇异的，利用它可以定义 $n \times n$ 矩阵

$$P_B^{(k)} = I - (B^{(k)})^T \left(B^{(k)}(B^{(k)})^T\right)^{-1} B^{(k)} \tag{5.28}$$

该矩阵具有性质

$$B^{(k)}P_B^{(k)} = 0, \quad P_B^{(k)}P_B^{(k)} = P_B^{(k)} \tag{5.29}$$

而且，对于任意向量 $d \in R^n$, $\hat{d} = P_B^{(k)}d$ 成为把 d 正投影到子空间 $\{y \in R^n \mid B^{(k)}y = 0\}$ 后的向量 (参照图 5.3)[1].

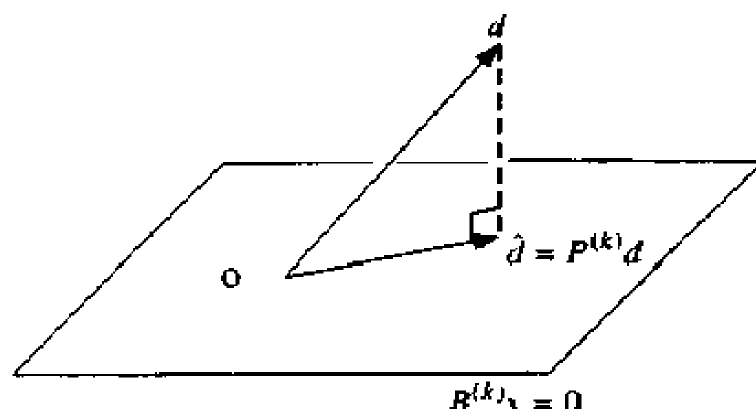


图 5.3 至子空间 $\{y \mid B^{(k)}y = 0\}$ 上的投影

1) 一般称具有这种性质的矩阵为投影矩阵 (projection matrix).

现在问题(5.22)的目标函数为分数形，所以先注意分子函数 $(c^{(k)})^T y$ 的最速下降方向 $-c^{(k)}$，考虑把它正投影到子空间 $\{y \in R^n \mid B^{(k)}y = 0\}$ 后的向量

$$-\hat{c}^{(k)} = -P_B^{(k)}c^{(k)} \tag{5.30}$$

根据式(5.29)、(5.30)，成立 $B^{(k)}\hat{c}^{(k)} = 0$，所以即使把点 e/n 沿向量 $-\hat{c}^{(k)}$ 的方向移动也不会违反问题(5.22)的等式约束条件(5.26)。因此Karmarkar法里，把向量 $-\hat{c}^{(k)}$ 正规化为长度是1的向量

$$d^{(k)} = -\frac{\hat{c}^{(k)}}{\|\hat{c}^{(k)}\|} \tag{5.31}$$

以此作为搜索方向，求出从点 e/n 出发，沿 $d^{(k)}$ 的方向前进步长 $\alpha > 0$ 所得到的点

$$y = \frac{1}{n}e + \alpha d^{(k)} \tag{5.32}$$

进一步，对点 y 实施投影变换 $T^{(k)}$ 的逆变换(5.19)，决定下面的点

$$x^{(k+1)} := (T^{(k)})^{-1}\left(\frac{1}{n}e + \alpha d^{(k)}\right) \tag{5.33}$$

容易验证这个点满足问题(5.13)的约束条件 $Ax = 0, e^T x = 1$. 另外，$x^{(k+1)}$ 是问题(5.13)的可行内点，所以只要选取使式(5.32)中 y 的各分量全部为正的步长 α 即可.

点 $x^{(k)}$ 经投影变换 $T^{(k)}$ 后恒映射至变量 y 的空间的单纯形重心，这个事实在选择步长时候起重要作用. 问题的最优解一般存在于单纯形的边界上，所以收敛于解的点列必然接近边界. 但是，在变量 x 的空间里当点 $x^{(k)}$ 在单纯形的边界附近时，从这个点出发沿目标函数的最速下降方向移动会马上

冲出边界．所以不能取太大的步长．但是，Karmarkar 法里，每次迭代中把点 $x^{(k)}$ 拉回到在变量 y 的空间里的单纯形重心 $e/n = T^{(k)}(x^{(k)})$，所以可以确保到单纯形的边界之间有足够的距离．这样一来，变量 y 的空间里，用式 (5.32) 决定下一步点的时候，可以选取在某种程度上较大的步长．

因为向量 $d^{(k)}$ 对应于针对问题 (5.22) 的目标函数之分子的最速下降方向，所以可能期待它也成为目标函数本身的下降方向．可实际上未必如此．因此 Karmarkar 法里，不考虑目标函数而是考虑下面的称为**势函数** (potential function) 的函数．通过生成使该函数单调减少的点列，得出到最优解的收敛 [1]．

$$f(x) = \ln\left(\frac{(c^T x)^n}{x_1 x_2 \cdots x_n}\right) = n \ln c^T x - \sum_{j=1}^{n} \ln x_j \tag{5.34}$$

根据关于问题 (5.13) 的假定 2，对所有的可行解 x 有 $c^T x > 0$，所以对于可行内点 $x > 0$ 一般成立 $c^T x > 0$，势函数 f 在可行内点的集合上的定义不会产生矛盾 [2]．进一步，若可行内点的序列 $\{x^{(k)}\}$ 满足

$$f(x^{(k)}) \longrightarrow -\infty \tag{5.35}$$

根据函数 f 的定义 (5.34)，成立 $c^T x^{(k)} \to 0$，所以可以认为点列 $\{x^{(k)}\}$ 收敛于问题 (5.13) 的最优解．

因为讨论非常复杂，所以仅仅列出结果．从点 $x^{(k)}$ 出发移动到式 (5.33) 的点 $x^{(k+1)}$ 时，如果选取步长 $\alpha = 1/(3n)$ 则可以断言

$$f(x^{(k+1)}) - f(x^{(k)}) < -\frac{1}{5} \tag{5.36}$$

---

1) 这里，ln 表示自然对数．

2) 对所有的可行解 x 恒成立 $c^T x = 0$ 的情况下无法定义函数 f，但是因为这种问题本来就无意义，所以不把它列为讨论对象．

成立．这样一来，固定步长为一定值 $\alpha = 1/(3n)$，反复进行上面的迭代，则对第 k 次迭代中势函数的值成立

$$f(x^{(k)}) < f(x^{(0)}) - \frac{1}{5}k \tag{5.37}$$

所以式 (5.35) 成立，也就知道点列 $\{x^{(k)}\}$ 收敛于问题 (5.13) 的最优解[3]．

根据上面的讨论知道，无限次进行 Karmarkar 法的迭代所生成的点列收敛于解．但是实际上，如下所述，可以断言有限次的迭代就足够了．作为表示问题 (5.13) 的数据大小的指标，和前节的椭球法情况下的式 (5.7) 同样定义

$$L = \sum_{i,j} \lceil \log(|a_{ij}| + 1) + 1 \rceil + \sum_{i} \lceil \log(|c_i| + 1) + 1 \rceil \tag{5.38}$$

现在，以 $x^{(0)} = e/n$ 为初始点执行 Karmarkar 法时，设目标函数值在第 k 次迭代里减小到满足

$$c^T x^{(k)} \leq 2^{-2L} c^T x^{(0)} \tag{5.39}$$

此时，把点 $x^{(k)}$ 的分量中与 0 充分接近者当作 0，则可以计算问题 (5.13) 的可行基解，它成为问题 (5.13) 的严密最优解．于是，Karmarkar 法中即使目标函数值 $c^T x$ 不严格为 0，也可以在满足式 (5.39) 的时刻结束迭代．

算法 KARMARKAR (Karmarkar 法)

第一步 (初始化)：取初始可行内点 $x^{(0)} := e/n$．令 $k := 0$．

---

1) 在各迭代里，也可以用适当的一维搜索方法求出使势函数减少最大的步长值来决定下一步的迭代点 $x^{(k+1)}$．当然，即使在这种情况下，也可以保证每次迭代里的势函数减少一定值 (1/5) 以上，式 (5.37) 成立．

第二步 (解的判定): 如果点 $x^{(k)}$ 满足不等式 (5.39) 则结束计算.

第三步 (下一步迭代点的计算): 根据式 (5.33) 计算 $x^{(k+1)}$. 令 $k := k+1$ 回到第二步

最后估计该算法的最大迭代次数. 首先, 根据式 (5.37) 和势函数的定义 (5.34), 对任意的 k 成立

$$n\ln c^T x^{(k)} - \sum_{j=1}^{n}\ln x_j^{(k)} \leq n\ln c^T x^{(0)} - \sum_{j=1}^{n}\ln x_j^{(0)} - \frac{1}{5}k \qquad (5.40)$$

然而, 单纯形 $S = \{x \in R^n \mid e^T x = 1, x \geq 0\}$ 上函数 $\sum_{j=1}^{n}\ln x_j$ 在 $x = e/n$ 时达到最大, 因为初始点实际上选了 $x^{(0)} = e/n$, 所以恒满足

$$\sum_{j=1}^{n}\ln x_j^{(k)} \leq \sum_{j=1}^{n}\ln x_j^{(0)}$$

因此, 由式 (5.40) 有

$$n\ln c^T x^{(k)} \leq n\ln c^T x^{(0)} - \frac{1}{5}k$$

也即成立

$$c^T x^{(k)} \leq \exp(-k/5n)\, c^T x^{(0)} \qquad (5.41)$$

这里, 把式 (5.39) 和 (5.41) 作一比较. 若在式 (5.41) 里令 $k > 10(\ln 2)nL$, 则式 (5.39) 可以得到满足. 所以知道, 最多在 $10(\ln 2)nL$ 次的迭代后, 算法的停止条件一定会得到满足. 也即, Karmarkar 法的迭代次数成为 $O(nL)$.

下面考虑算法整体的计算量. Karmarkar 法的各次迭代里需要计算量最多的是向量 $c^{(k)}$ 的正投影计算 (5.30), 它可归结为解以 $B^{(k)}(B^{(k)})^T$ 为系数矩阵的联立一次方程组. 老老实实进行

这个计算的话，因为求矩阵 $B^{(k)}(B^{(k)})^T$ 的分量要 $O(nm^2)$，解联立一次方程组要 $O(m^3)$ 的计算量。如果考虑到 $n > m$，一次迭代里的计算量成为 $O(nm^2)$。实际上，$B^{(k)}(B^{(k)})^T = A(D^{(k)})^2A^T$，其中每次迭代发生变化的仅仅是对角矩阵 $D^{(k)}$ 的分量。利用这个事实有可能稍微减少一点每次迭代的计算量。

以上关于 Karmarkar 法计算量的讨论可以概括为如下定理。

定理 5.2 (Karmarkar 法的多项式性) 针对问题 (5.13) 的 Karmarkar 法在 $O(nL)$ 次迭代后一定停止。另外，每次迭代的计算量可以控制在问题规模的多项式阶里，所以 Karmarkar 法是多项式时间算法。

上面关于 Karmarkar 法的计算量的估计从实用观点来看决不能说是小的，如果实际上真需要那么些计算量则不可能战胜单纯形法。Karmarkar 法为代表的内点法之所以引人注目，是因为实际上进行计算的时候，迭代次数不太依赖于问题的规模。对一般的问题在数十次的迭代里就可得到十分满意的近似解。这是个令人吃惊的经验事实。

5.3 仿射变换法

为了使用问题 (5.13) 的投影变换，Karmarkar 法的描述显得有些复杂，问题的设定也留有不自然的印象。为克服这个缺点，**仿射变换法** (affine scaling method) 不用投影变换而用仿射变换 (线性变换)。它具有不需 Karmarkar 法那样的特别假定就能执行的优点。仿射变换法是六十年代由 I. I. Dikin (И. И. Дикин) 提出的方法，当时不怎么引人注目，但是在

Karmarkar 法发表以后再次发掘，这一研究便活跃起来了. 该方法尚未像 Karmarkar 法那样被证明为多项式时间算法，但是根据数值实验，人们发现它的计算效率相当好.

考虑下面的标准形线性规划问题.

$$
\begin{aligned}
&\text{目标函数：}\quad c^T x \longrightarrow \text{最小} \\
&\text{约束条件：}\quad Ax = b,\ x \geq 0
\end{aligned}
\tag{5.42}
$$

这里，变量 x 是 n 维向量，A 是 $m \times n$ 常数矩阵，b 和 c 分别是 m 维及 n 维常数向量. 进一步，假定矩阵 A 的秩为 m. 另外，虽然没有像 Karmarkar 法那样假定已经知道目标函数的最小值，但是作为初始点 $x^{(0)}$，假设至少知道一个满足

$$
Ax = b,\ x > 0 \tag{5.43}
$$

的点. 和前节同样，这种点称为**可行内点** (feasible interior point)

仿射变换法也和 Karmarkar 法同样，是从可行内点 $x^{(0)}$ 开始，生成收敛于问题最优解的可行内点的序列 $\{x^{(k)}\}$ 的方法. 现假设给定可行内点 $x^{(k)}$，说明如何确定下一步的点 $x^{(k+1)}$.

对于给定的点 $x^{(k)} > 0$，定义一个把任意点 $x > 0$ 仍然映射为所有分量为正的点 $y > 0$ 的映射

$$
y_j = x_j / x_j^{(k)} \quad (j = 1, 2, \ldots, n) \tag{5.44}
$$

和前节一样，把以 $x_j^{(k)}$ 为对角分量的对角矩阵记为 $D^{(k)}$ (参照式 (5.17))，式 (5.44) 可以用下面的向量来表示.

$$
y = (D^{(k)})^{-1} x \tag{5.45}
$$

它是变量 x 的空间到变量 y 的空间的线性变换，请特别注意它把现在的点 $x^{(k)}$ 映射至点 $e = (1, 1, \ldots, 1)^T$.

利用式 (5.45) 的变换，可以把问题 (5.42) 表示为下面的等价问题.

$$
\begin{array}{ll}
\text{目标函数：} & (c^{(k)})^T y \quad \rightarrow \text{ 最小} \\
\text{约束条件：} & A^{(k)} y = b,\ y \geq 0
\end{array}
\tag{5.46}
$$

这里，$c^{(k)}$, $A^{(k)}$ 分别是由和式 (5.23), (5.24) 一样的

$$c^{(k)} = D^{(k)} c \tag{5.47}$$

$$A^{(k)} = AD^{(k)} \tag{5.48}$$

所定义的 n 维向量和 $m \times n$ 矩阵.

问题 (5.46) 里，考虑从对应现在点 $x^{(k)}$ 的点 $y = e$ 开始，向使目标函数 $(c^{(k)})^T y$ 减少最大的方向移动. 这个时候，在不违反约束 $A^{(k)} y = b$ 的条件下，定义到子空间 $\{y \in R^n \mid A^{(k)} y = 0\}$ 上的投影矩阵

$$P_A^{(k)} = I - (A^{(k)})^T \left(A^{(k)} (A^{(k)})^T \right)^{-1} A^{(k)} \tag{5.49}$$

把目标函数的最速下降方向向量 $-c^{(k)}$ 投影到子空间 $\{y \in R^n \mid A^{(k)} y = 0\}$ 上得到的向量

$$-\hat{c}^{(k)} = -P_A^{(k)} c^{(k)} \tag{5.50}$$

进一步作长度为 1 的正规化

$$\hat{d}^{(k)} = -\frac{\hat{c}^{(k)}}{\|\hat{c}^{(k)}\|} \tag{5.51}$$

以它为搜索方向 (移动方向). 其次，求出从点 $y = e$ 出发沿向量 $\hat{d}^{(k)}$ 的方向前进步长 $0 < \alpha < 1$ 后得到的点

$$y = e + \alpha \hat{d}^{(k)} \tag{5.52}$$

进一步，通过对点 y 实施式 (5.45) 的逆变换，确定变量 x 空间里的下一步迭代点

$$x^{(k+1)} = D^{(k)}\left(e + \alpha\hat{d}^{(k)}\right) = x^{(k)} + \alpha D^{(k)}\hat{d}^{(k)} \tag{5.53}$$

为求出搜索方向 $\hat{d}^{(k)}$，有必要计算式 (5.50) 的向量 $\hat{c}^{(k)}$ 实际上如下的考虑方法很方便. 首先，注意到根据式 (5.47) (5.50)，成立

$$\hat{c}^{(k)} = D^{(k)}\left(c - A^T\left(A\left(D^{(k)}\right)^2A^T\right)^{-1}A\left(D^{(k)}\right)^2c\right) \tag{5.54}$$

这里，如果令

$$r^{(k)} = c - A^T\left(A\left(D^{(k)}\right)^2A^T\right)^{-1}A\left(D^{(k)}\right)^2c \tag{5.55}$$

则由式 (5.54) 有

$$\hat{c}^{(k)} = D^{(k)}r^{(k)} \tag{5.56}$$

进一步令

$$w^{(k)} = \left(A\left(D^{(k)}\right)^2A^T\right)^{-1}A\left(D^{(k)}\right)^2c \tag{5.57}$$

则式 (5.55) 可表示为

$$r^{(k)} = c - A^Tw^{(k)} \tag{5.58}$$

于是，首先求出式 (5.57) 的 $w^{(k)}$，把它代入式 (5.58) 计算 $r^{(k)}$，最后由式 (5.56) 求出 $\hat{c}^{(k)}$ 即可.

这里，确认一下点 $x^{(k+1)}$ 是问题 (5.42) 的可行内点，以及根据式 (5.53) 的迭代得到的目标函数值会减少.

首先，由式 (5.48)，(5.53) 成立有

$$Ax^{(k+1)} = Ax^{(k)} + \alpha A^{(k)}\hat{d}^{(k)} \tag{5.59}$$

然而根据投影矩阵的性质有 $A^{(k)}P_A^{(k)} = 0$、所以如果考虑式 (5.50)、(5.51)，则式 (5.59) 的右边第二项成为 0. 于是，由假定有 $Ax^{(k)} = b$、因而得到 $Ax^{(k+1)} = b$. 另外，搜索方向向量 $d^{(k)}$ 已作了长度为 1 的正规化，因步长 $0 < \alpha < 1$，故式 (5.52) 的 y 的分量全部为正[1]. 因此，对它进行式 (5.45) 的逆变换得到的点 $x^{(k+1)}$ 也满足 $x^{(k+1)} > 0$.

下面，根据式 (5.53) 有

$$c^T x^{(k)} - c^T x^{(k+1)} = -\alpha c^T D^{(k)} \hat{d}^{(k)}$$

再注意到由式 (5.47)、(5.50)、(5.51) 以及投影矩阵的性质成立 $P_A^{(k)} = (P_A^{(k)})^2$，则上式可以改写为

$$c^T x^{(k)} - c^T x^{(k+1)} = \frac{\alpha (c^{(k)})^T P_A^{(k)} c^{(k)}}{\|P_A^{(k)} c^{(k)}\|} = \alpha \|P_A^{(k)} c^{(k)}\| = \alpha \|\hat{c}^{(k)}\| \tag{5.60}$$

该式右边为正，所以各迭代里目标函数值 $c^T x^{(k)}$ 单调减少.

上面所述的仿射变换法算法可以概括如下. 针对该算法没有像 Karmarkar 法那样在理论上的明确的终止准则，所以在某次迭代里，若目标函数值的减少量足够小，便结束计算.

算法 AFFINE (仿射变换法)

第一步 (初始化): 选取适当的初期可行内点 $x^{(0)}$，步长 $0 < \alpha < 1$，足够小的 $\varepsilon > 0$. 令 $k := 0$.

第二步 (计算下一步迭代点): 根据式 (5.53) 计算 $x^{(k+1)}$.

---

1) 实际上，如果式 (5.52) 中的点 y 在满足 $y > 0$ 的范围内，步长 α 比 1 大也无妨. 比如，采用求出每次迭代中使式 (5.52) 右边为正的 α 的上限值，(为使得到的点确实成为可行内点) 把它稍微缩小 (比如 0.99 倍) 得到的值作为步长的方法，在实际上很有效.

第三步 (解的判定): 如果 $c^Tx^{(k)} - c^Tx^{(k+1)} < \varepsilon$ 则结束计算. 否则, 令 $k := k+1$ 回到第二步.

仿射变换法尚未像 Karmarkar 法那样被证明为多项式时间算法. 但是, 不考虑算法第三步的停止条件, 而让迭代无限重复下去, 则可以证明生成的点列在适当的条件下收敛于问题的最优解.

为此, 导入在第 1 章里考察单纯形法时用到的非退化假定 (nondegeneracy assumption), 也即问题 (5.42) 的任意可行基解里基底变量的值全部为正的假定 (参照 1. 4). 另一方面, 针对问题 (5.42) 的对偶问题

$$\begin{aligned} &\text{目标函数:} \quad b^Tw \longrightarrow \text{最大} \\ &\text{约束条件:} \quad A^Tw \leq c \end{aligned} \tag{5.61}$$

其中的 n 个不等式约束里, 假定对任意的 w 同时成立等式的个数最多 [1] 是 m. 该假定对应于针对对偶问题的非退化假定, 称为 **对偶非退化** (dual nondegeneracy) 假定. (与此相对, 针对问题 (5.42) 的非退化假定称为 **原非退化** (primal nondegeneracy) 假定.)

这里考虑由仿射变换法生成的点列 $\{x^{(k)}\}$, 在上面的假定下, 可以断言对正好 m 个下标 j 有 $x_j^{(k)} \not\to 0$, 对 $n-m$ 个下标 j 有 $x_j^{(k)} \to 0$. 这意味着, 以对应于前者的下标的变量为基底变量, 后者为非基底变量的可行基解成为点列 $\{x^{(k)}\}$ 的极限, 但是实际上, 可以证明该可行基解为问题 (5.42) 的最优解. 进一步还可以断言, 式 (5.57) 给出的向量序列 $\{w^{(k)}\}$ 收敛于对偶问题 (5.61) 的最优解.

---

1) 几何学上, 考虑变量 w 的空间 (m 维空间 R^m) 里不等式约束条件 $A^Tw \leq c$ 分别对应的 n 个等式所确定的超平面的时候, 假定没有其中任意 $m+1$ 个以上相交于一点的情况.

定理 5.3 (仿射变换法的全局收敛性) 假定问题 (5.42) 有最优解，而且满足原非退化以及对偶非退化假定. 在 $\epsilon = 0$ 的时候，仿射变换法生成的无限点列 $\{x^{(k)}\}$ 收敛于问题 (5.42) 的最优可行基解，点列 $\{w^{(k)}\}$ 收敛于对偶问题 (5.61) 的最优解.

非退化假定对现实问题来说是相当严格的条件. 有人尝试去掉该假定证明仿射变换法在理论上的全局收敛性. 实际上，如果设定步长 α 为适当值，则即使没有非退化假定也证明了 $\{x^{(k)}\}$ 收敛于最优解.

并不限于仿射变换法，关于一般的内点法，为了保证全局收敛性和多项式时间性等理论上的收敛性，步长一般都必须设定得相当小. 但是，根据数值实验等经验，人们知道在不违背生成点列的可行性条件范围内尽量选取大步长的话，在实际计算效率方面更令人满意. 这样，关于内点法的步长之选择，理论上未知部分还有不少.

5.4 减势法

虽然前节的仿射变换法实际上有效，但是关于计算复杂性，在理论上还没有证明其多项式性. 这节说明与 Karmarkar 法类似的另一个内点法. 通过利用势函数来决定搜索方向及步长，成功保证了多项式性.

和前节同样，要处理的问题是下面的标准形问题.

$$\begin{aligned} &\text{目标函数:} \quad c^T x \longrightarrow \text{最小} \\ &\text{约束条件:} \quad Ax = b,\ x \geq 0 \end{aligned} \tag{5.62}$$

该问题的对偶问题由

$$\text{目标函数：}\quad b^T w \longrightarrow \text{最大}$$
$$\text{约束条件：}\quad A^T w \leq c$$

给出 (参照 1. 7 节). 但是为了以下的讨论，使用松驰变量把它改写为

$$\begin{aligned}&\text{目标函数：}\quad b^T w \longrightarrow \text{最大}\\&\text{约束条件：}\quad A^T w + s = c,\quad s \geq 0\end{aligned} \tag{5.63}$$

这里，与问题 (5.62), (5.63) 相关联，定义下面的函数.

$$f(x,s) = \rho \ln x^T s - \sum_{j=1}^{n} \ln x_j s_j \tag{5.64}$$

这里，$\rho = n + \sqrt{n}$ (n 是向量 x 的维数), $s = (s_1, s_2, \ldots, s_n)^T$ 是问题 (5 63) 里出现的松驰变量. 称该函数为针对问题 (5.62) 的**原对偶势函数** (primal-dual potential function).

因为对于问题 (5.62), (5.63) 的任意可行解 x, w, s 成立

$$x^T s = x^T (c - A^T w) = c^T x - b^T w \tag{5.65}$$

所以原对偶势函数 $f(x,s)$ 也可以写为

$$f(x,s) = \rho \ln (c^T x - b^T w) - \sum_{j=1}^{n} \ln x_j - \sum_{j=1}^{n} \ln s_j \tag{5.66}$$

把式 (5.66) 的势函数和针对 Karmarkar 法的势函数 (5.34) 比较一下，除去式 (5.66) 里包含有对偶问题的数据这点以外，两者极其相似.

然而，因为 $\rho = n + \sqrt{n}$, 把式 (5.64) 变形为

$$f(x,s) = \sqrt{n} \ln x^T s - \sum_{j=1}^{n} \ln \left(\frac{x_j s_j}{x^T s} \right) \tag{5.67}$$

根据算术平均和几何平均不等式

$$\frac{x^T s}{n} = \frac{\sum_{j=1}^{n} x_j s_j}{n} \geq \sqrt[n]{\prod_{j=1}^{n} x_j s_j}$$

也即

$$\frac{\prod_{j=1}^{n} x_j s_j}{(x^T s)^n} \leq n^{-n}$$

所以由式 (5.67), 成立有

$$f(x, s) = \sqrt{n} \ln x^T s - \ln \frac{\prod_{j=1}^{n} x_j s_j}{(x^T s)^n} \geq \sqrt{n} \ln x^T s + n \ln n > \sqrt{n} \ln x^T s \tag{5.68}$$

这里, 假设对于原问题 (5.62), 对偶问题 (5.63) 各自的可行内点序列 $\{x^{(k)}\}$, $\{(w^{(k)}, s^{(k)})\}$ 成立

$$f(x^{(k)}, s^{(k)}) \to -\infty$$

这个时候, 根据式 (5.65) 和 (5.68),

$$c^T x^{(k)} - b^T w^{(k)} \longrightarrow 0$$

于是, 根据线性规划法的对偶理论 (参照定理 1.4 (iv)), 点列 $\{x^{(k)}\}$ 和 $\{(w^{(k)}, s^{(k)})\}$ 分别收敛于原问题 (5.62) 和对偶问题 (5.63) 的最优解. 实际上人们知道和 Karmarkar 法同样, 当 $c^T x^{(k)} - b^T w^{(k)}$ 足够接近 0 而且满足

$$c^T x^{(k)} - b^T w^{(k)} < 2^{-L} \tag{5.69}$$

则从该点出发可以计算问题的最优解. 这里, L 是表示问题数据大小的参数 (参照式 (5.38)). 另外, 如果考虑式 (5.68), 则式 (5.69) 在

$$f(x^{(k)}, s^{(k)}) < -\sqrt{n} L \tag{5.70}$$

时成立.

减势法 (potential reduction method) 是通过让原对偶势函数 $f(x,s)$ 单调减少下去，最终使式 (5.70) (也即，式 (5.69)) 得以满足从而求出解的迭代法. 现在，给定了第 k 次迭代里满足

$$Ax^{(k)} = b, \quad x^{(k)} > 0, \quad s^{(k)} = c - A^T w^{(k)} > 0 \tag{5.71}$$

的原问题 (5.62), 对偶问题 (5.63) 的可行内点 $x^{(k)}, w^{(k)}, s^{(k)}$. 现在，回忆前节的仿射变换法是利用向量

$$\hat{c}^{(k)} = \left(I - (A^{(k)})^T \left(A^{(k)}(A^{(k)})^T\right)^{-1} A^{(k)}\right) D^{(k)} c \tag{5.72}$$

由

$$x^{(k+1)} = x^{(k)} - \alpha D^{(k)} \frac{\hat{c}^{(k)}}{\|\hat{c}^{(k)}\|} \tag{5.73}$$

决定下一步迭代点 (参照式 (5.51), (5.53)) 这里，和式 (5.17), (5.24) 一样，$D^{(k)}$ 为以 $x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}$ 为对角分量的对角矩阵，$A^{(k)}$ 是由 $A^{(k)} = AD^{(k)}$ 定义的矩阵. 与此相对，减势法里，利用把式 (5.72) 右边的向量 c 替换为原对偶势函数 $f(x,s)$ 关于变量 x 的梯度 $\nabla_x f(x^{(k)}, s^{(k)})$ 后得到的向量

$$\hat{p}^{(k)} = \left(I - (A^{(k)})^T \left(A^{(k)}(A^{(k)})^T\right)^{-1} A^{(k)}\right) D^{(k)} \nabla_x f(x^{(k)}, s^{(k)}) \tag{5.74}$$

来确定搜索方向. 根据原对偶势函数的定义 (5.64) 有

$$\nabla_x f(x^{(k)}, s^{(k)}) = \frac{\rho}{(x^{(k)})^T s^{(k)}} s^{(k)} - (D^{(k)})^{-1} e$$

(e 是分量全部为 1 的向量), 注意到 $s^{(k)} = c - A^T w^{(k)}$ 和 $A^{(k)} = AD^{(k)}$, 式 (5.74) 经整理后可以得到

$$\hat{p}^{(k)} = \left(I - (A^{(k)})^T \left(A^{(k)}(A^{(k)})^T\right)^{-1} A^{(k)}\right) \left(\frac{\rho}{(x^{(k)})^T s^{(k)}} D^{(k)} c - e\right) \tag{5.75}$$

进一步，减势法里，判定向量 $\hat{p}^{(k)}$ 的范数是否比某个常数 $\gamma > 0$ 更大，仅在

$$||\hat{p}^{(k)}|| > \gamma \tag{5.76}$$

成立时更新迭代点 $x^{(k)}$,

$$x^{(k+1)} := x^{(k)} - \alpha \frac{D^{(k)}\hat{p}^{(k)}}{||\hat{p}^{(k)}||} \tag{5.77}$$

这里，$0 < \alpha < 1$ 是步长. 进一步，这种情况下，对偶变量的值

$$w^{(k+1)} := w^{(k)}, \quad s^{(k+1)} := s^{(k)} \tag{5.78}$$

无变更.

相反，式 (5.76) 不成立时，不更新原变量

$$x^{(k+1)} := x^{(k)} \tag{5.79}$$

而对对偶变量作如下更新

$$\begin{aligned} w^{(k+1)} &:= \left(A^{(k)}(A^{(k)})^T\right)^{-1} A^{(k)} \left(D^{(k)}c - \frac{(x^{(k)})^T s^{(k)}}{\rho} e\right) \\ &= \left(A(D^{(k)})^2 A^T\right)^{-1} \left(A(D^{(k)})^2 c - \frac{(x^{(k)})^T s^{(k)}}{\rho} b\right) \end{aligned} \tag{5.80}$$

$$s^{(k+1)} := c - A^T w^{(k+1)} \tag{5.81}$$

另外，容易验证式 (5.81) 的 $s^{(k+1)}$ 和式 (5.75) 的 $\hat{p}^{(k)}$ 之间成立有

$$\hat{p}^{(k)} = \frac{\rho}{(x^{(k)})^T s^{(k)}} D^{(k)} s^{(k+1)} - e \tag{5.82}$$

的关系，所以计算 $\hat{p}^{(k)}$ 时，首先从式 (5.80), (5.81) 中求出 $s^{(k+1)}$, 把它代入式 (5.82) 中求出 $\hat{p}^{(k)}$. 这样就没有必要在做式 (5.76) 的判定后重新计算 $s^{(k+1)}$.

算法 PR (减势法)

第一步 (初始化): 选取适当的初始可行内点 $x^{(0)}, w^{(0)}, s^{(0)}$, 步长 $0 < \alpha < 1$, 参数 $\gamma > 0$. 令 $k := 0$.

第二步 (计算下一步迭代点): 计算式 (5.75) 所定义的向量 $\hat{p}^{(k)}$. 如果式 (5.76) 成立, 则由式 (5.77), (5.78) 确定 $x^{(k+1)}, w^{(k+1)}, s^{(k+1)}$. 否则, 由式 (5.79)-(5.81) 确定 $x^{(k+1)}, w^{(k+1)}, s^{(k+1)}$.

第三步 (解的判定): 如果满足 $c^T x^{(k+1)} - b^T w^{(k+1)} < 2^{-L}$, 则结束计算. 否则, 令 $k := k+1$ 回到第二步.

下面确认算法的第二步所确定的 $x^{(k+1)}, w^{(k+1)}, s^{(k+1)}$ 为可行内点. 首先, $x^{(k+1)}$ 由式 (5.77) 给出的情况下, 根据前节与针对仿射变换法的讨论 (参照式 (5.59) 及其后的段落) 一样的理由成立

$$Ax^{(k+1)} = b, \quad x^{(k+1)} > 0$$

其次, 考虑 $w^{(k+1)}$ 及 $s^{(k+1)}$ 由式 (5.80), (5.81) 给出的情况. 为证明它是可行内点只要证明 $s^{(k+1)} > 0$ 就够了. 因此, 如果向量 $s^{(k+1)}$ 具有满足 $s_\ell^{(k+1)} \le 0$ 的分量, 则根据式 (5.82), 对于向量 $\hat{p}^{(k)}$ 的对应分量成立 $\hat{p}_\ell^{(k)} \le -1$. 进一步, 由范数的定义, 成立

$$\|\hat{p}^{(k)}\| = \sqrt{\sum_{j=1}^{n} \left(\hat{p}_j^{(k)}\right)^2} \ge |\hat{p}_\ell^{(k)}| \ge 1$$

在更新 $w^{(k+1)}$ 和 $s^{(k+1)}$ 时式 (5.76) 的反向不等式 $\|\hat{p}^{(k)}\| \le \gamma < 1$ 应该成立. 这导致矛盾, 所以, 必须有 $s^{(k+1)} > 0$.

作为更重要的事实, 如适当设定在判定 $\hat{p}^{(k)}$ 的大小时 (式 (5.76)) 用到的参数 γ, 则在满足式 (5.76) 的时候, 通过巧妙选取步长 α, 可成立

$$f(x^{(k)} - \alpha D^{(k)}\hat{p}^{(k)}/\|\hat{p}^{(k)}\|, s^{(k)}) < f(x^{(k)}, s^{(k)}) - \delta \qquad (5.83)$$

在不满足式 (5.76) 的时候，对式 (5.81) 的 $s^{(k+1)}$ 可以断言成立

$$f(x^{(k)}, s^{(k+1)}) < f(x^{(k)}, s^{(k)}) - \delta \tag{5.84}$$

这里， δ 是依赖于 γ 和 α 而确定的正常数[1)]．于是，根据式 (5.77) (5.81)，不论在哪种情况下都成立

$$f(x^{(k+1)}, s^{(k+1)}) < f(x^{(k)}, s^{(k)}) - \delta \tag{5.85}$$

利用式 (5.85) 可以估计减势法的迭代次数．为此要作的假定是选取初期可行内点使之满足

$$f(x^{(0)}, s^{(0)}) \leq O(\sqrt{n}L) \tag{5.86}$$

实际上选取成立式 (5.86) 的初始点时需要些技巧，在这里仅仅指出只要选取初始点使之满足

$$x_1^{(0)} s_1^{(0)} = x_2^{(0)} s_2^{(0)} = \cdots = x_n^{(0)} s_n^{(0)}$$

以及

$$(x^{(0)})^T s^{(0)} \leq 2^L$$

则根据式 (5.67)，条件 (5.86) 确实成立．式 (5.86) 意味着对某个常数 $\mu > 0$ 有

$$f(x^{(0)}, s^{(0)}) \leq \mu\sqrt{n}L$$

所以由式 (5.85)，第 k 次迭代里成立

$$f(x^{(k)}, s^{(k)}) < f(x^{(0)}, s^{(0)}) - k\delta \leq \mu\sqrt{n}L - k\delta$$

---

1) $\gamma = \min\left(\beta\sqrt{n/(n+\beta^2)}, 1 - \beta\right)$ (这里 $\beta = 0.43$)，如果令 $\alpha = 0.3$，则对 $\delta = 0.05$ 式 (5.83)，(5.84) 成立

于是，这里令 $k=(\mu+1)\delta^{-1}\sqrt{n}L$ 则

$$f(x^{(k)},s^{(k)}) < -\sqrt{n}L$$

也即式 (5.70) 成立，而且这个时候第三步的停止条件也得以满足，所以知道算法的最大迭代次数是 $O(\sqrt{n}L)$.

定理 5.4 (减势法的多项式性) 在选取初始点使之满足式 (5.86) 的时候，针对问题 (5.62) 的减势法在 $O(\sqrt{n}L)$ 次的迭代里停止. 另外，各迭代里的计算量可以控制在问题规模的多项式内，所以减势法是多项式时间算法.

5.5 文献及其它话题

本章叙述了针对线性规划问题的新方法 — 椭球法及内点法. 比起单纯形法这两个方法历史都短，详细讲解的教科书并不多. 在第 1 章介绍的线性规划法教科书里， Chvátal [C83] 讲解了椭球法，今野 [K87] 讲解了内点法，伊理 [I86] 讲解了两个方法. 另外， Goldfarb, Todd [GT89] 里也有综合讲解.

5.1 节的椭球法是 Khachiyan [K79] 提出的针对线性规划问题的第一个多项式时间算法，发表的当时引起了巨大反响. 椭球法并不是 Khachiyan 一人的创作，而是发展了在此之前原苏联科学家勤勤恳恳持之以恒研究出的非线性最优化方法的结果. Bland 等的综述论文 [BGT81] 里有关于椭球法的详细讲解.

5.2 节所述的方法是 Karmarkar [K84] 在 1984 年发表的多项式时间算法. 它成为其后有关内点法的活跃研究之出发点. 5.3 节提到的仿射变换法在 Karmarkar 的论文发表之后，立刻由 Barnes [B86] 等做了更进一步的研究，但是后来判明 Dikin

[D67] 早在六十年代就提出了同样的方法. 仿射变换法还没有被证明是多项式时间算法. 另外, 全局收敛性的证明在非退化的假定下比较容易, 但在出现退化的情况下有必要作复杂讨论(土屋 [T91]). 有些研究者在尝试把仿射变换法与势函数结合起来构造多项式时间算法, 5.3 节的结果引自 Ye [Y91].

内点法里除上面所述的内容之外还可以考虑各种各样的变化 [GT89]. 另外, 虽然本书几乎没能涉及到, 为有效执行内点法, 有必要在包含用到的数据结构在内的各种计算上下工夫 [AKRV89, M92]. 然而, 像在 5.2 节最后讲到的那样, 有经验事实表明, 内点法与问题的大小无关, 几乎在一定的迭代次数里可以得到足够精度的近似解. 于是, 在与单纯形法作比较的情况下, 可以认为随着问题的规模变大, 内点法越来越有利. 实际上, 也有计算实验结果 [MMS89,MM87] 报告说即使是数百变量程度的问题, 内点法比起单纯形法也要快. 另外, Bixby 等 [BGLMS92] 报告了把 [LMS92] 提出的内点法与单纯形法进行组合, 解决了变量数目达到 12,753,313 的巨大规模问题的事例.

把内点法应用到不限于线性规划问题的更一般的问题的尝试很活跃. 特别是, 小岛等 [KMNY91] 把内点法推广到包含二次规划问题的称为**线性互补问题** (linear complementarity problem) 的问题里, 成功地构造了具有和线性规划问题情况同样程度的计算量的多项式时间算法.

第 6 章 神经网络和最优化

神经网络是模仿人脑结构进行高度平行分散处理的数学模型. 神经网络的想法本身并不是什么新东西, 可最近有人提出了几个有意义的想法, 掀起了新的研究热潮. 这章从被称为神经网络雏形的古典模式识别开始到误差逆传播法及 Hopfield 网络产生的组合优化法, 选取与最优化密切相关的课题进行讲解.

6.1 神经网络

一般说来, 神经网络用连结着神经细胞的神经回路网经模型化后成为图 6.1 所示的有向图来表示. 神经网络的各节点 (神经元) 是处理要素, 该处接受其它节点的输出信号作为输入. 进一步对这些输入作适当变换后决定该节点的输出, 并传递给其它节点. 所有这些操作都是各节点以平行分散的方式进行处理.

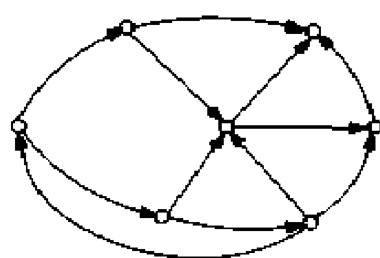


图 6.1 神经网络

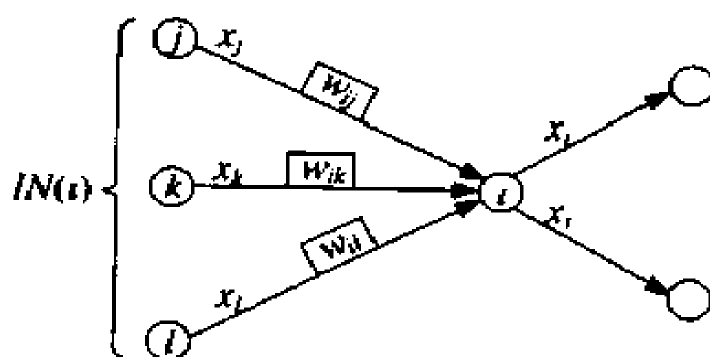
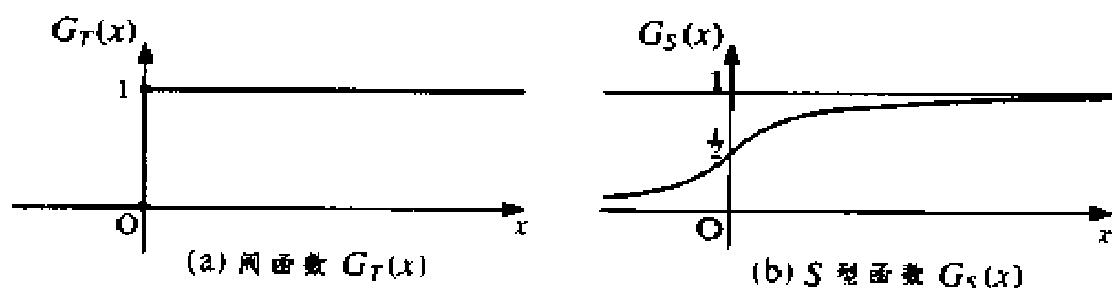


图 6.2 节点 i 的输入和输出

现在在某时刻节点 i 的输出设为 x_i. 这里 x_i 是满足 $0 \leq x_i \leq 1$ 的实数. 再把能够直接传送输出信号到节点 i 的节点 j 的集合用 $IN(i)$ 表示. 这时若从节点 $j \in IN(i)$ 送来的信号 x_j 到达节点 i 时扩大 w_{ij} 倍，则节点 i 的输入的总和为 $\sum_{j \in IN(i)} w_{ij} x_j$ (图 6.2). 节点 i 把这个输入用适当的输出函数 G: $R \to [0,1]$ 变换成

$$x_i = G\left(\sum_{j \in IN(i)} w_{ij} x_j - \theta_i \right) \tag{6.1}$$



(a) 阈函数 $G_T(x)$　　(b) S 型函数 $G_S(x)$

图 6.3　输出函数 $G(x)$ 之例

参数 w_{ij} 是表示节点 j 和 i 连接强度的权系数 [1]. 式 (6.1) 的 θ_i 是表示节点 i 的行动基准的参数，称为**阈值**(threshold value). 另外作为函数 G, 经常用到图 6.3 所示的 **阈函数** (threshold function)

$$G_T(x) = \begin{cases} 1, & \text{当} \quad x > 0 \quad \text{时} \\ 0 \text{ 或 } 1, & \text{当} \quad x = 0 \quad \text{时} \\ 0, & \text{当} \quad x > 0 \quad \text{时} \end{cases} \tag{6.2}$$

以及 **S- 型函数** (sigmoid function)

$$G_S(x) = \frac{1}{1 + \exp(-x)} \tag{6.3}$$

特别是 S- 型函数 G_S 可微，其导函数 G'_S 成立有

$$G'_S(x) = \frac{\exp(-x)}{(1 + \exp(-x))^2} = G_S(x)(1 - G_S(x)) \tag{6.4}$$

1) 权系数 w_{ij} 并不限于正.

在这种神经网络里，要解决的问题是，为了让网络作为整体完成所要求的操作该如何确定节点间的权系数以及各节点的阈值．下面要讲，人们为此考虑了各种基于学习的方法．

6.2 模式识别和感知器

6.2.1 模式识别

模式识别 (pattern recognition) 的基本问题是，在给定图像和声音等的模式时，识别它们是否属于某个类别．要是把各模式用 m 维向量表示的话，可把给定的模式的集合看成是 m 维空间 R^m 上的点集合．这时，有像图 6.4(a) 那样，属于某个类别的模式 (○) 和不属于的模式 (×) 可以用空间 R^m 的某个超平面分离的情况，也有像图 6.4(b) 那样不可能用超平面分离的情况．在前面那种情况下，称所给的模式集合 **线性可分** (linearly separable)．

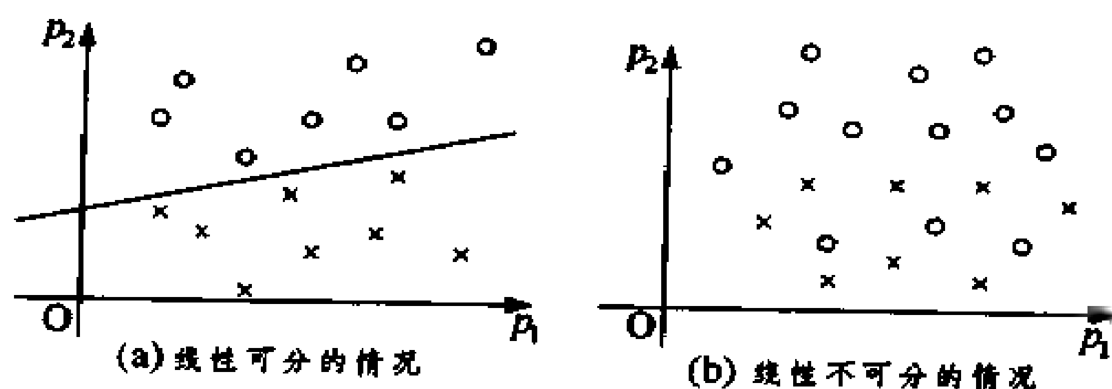


图 6.4　模式空间

现在，给定 N 个模式向量 $p^1, p^2, \ldots, p^N$，不失一般性，设这些模式向量中的先头 N' 个属于某个类，而剩下的 $N-N'$ 个不属于该类．这时，若是这个模式集合线性可分，那么存在 m 维向量 $v=(v_1, v_2, \ldots, v_m)$ 及实数 θ 使得

$$
\begin{cases} v^T p^s - \theta > 0 & (s = 1, 2, \ldots, N') \\ v^T p^s - \theta < 0 & (s = N'+1, N'+2, \ldots, N) \end{cases} \tag{6.5}
$$

成立. 也就是说, 模式向量 p 的空间 R^m 中用 $\{p | v^T p = \theta\}$ 表示的超平面可以分离属于这个类的模式和不属于这个类的模式. 于是, 对于线性可分的模式集合来说, 若是知道满足式 (6.5) 的 $(v, \theta) \in R^{m+1}$, 则图像模式的识别实际上就成为可能的了.

因此, 若把 $m+1$ 维向量 $(p^s, -1)$ 以及 (v, θ) 分别记为 q^s, w, 则 (6.5) 可写成

$$
\begin{cases} w^T q^s > 0 & (s = 1, 2, \ldots, N') \\ w^T q^s < 0 & (s = N'+1, N'+2, \ldots, N) \end{cases} \tag{6.6}
$$

以下考虑 (6.6) 而非 (6.5). 用满足 (6.6) 的向量 w 表示分离模式向量 q 的空间的超平面 $w^T q = 0$. 进一步, 为表述方便, 把向量 w, q 的维数 $m+1$ 记为 n.

但是, 把满足式 (6.6) 的 w 扩大任意正数倍后仍满足式 (6.6), 因此可把式 (6.6) 换成

$$
\begin{cases} w^T q^s \geq 1 & (s = 1, 2, \ldots, N') \\ w^T q^s \leq -1 & (s = N'+1, N'+2, \ldots, N) \end{cases} \tag{6.7}
$$

与满足式 (6.6) 的全体 w 的集合是开集合相比, 满足式 (6.7) 的是闭集合, 在很多场合下后者的数学处理要方便些.

6.2.2 感知器

感知器 (perceptron) 是用于认识模式的"计算机器". 自五十年代后期至六十年代引起了广泛注意. 特别是, 对上述问题用上了图 6.5 所示的, 称为基本感知器的神经网络. 在图 6.5 的基本感知器里, 网络的输入是 n 维向量 x, 其各分量 x_j 是通过节

点 j 送到节点 r 的．现在如果各节点 $j=1,2,\ldots,n$ 和节点 r 的权系数为 w_j，节点 j 的出力信号 x_j 的加权和 $w^Tx=\sum_{j=1}^{n}w_jx_j$ 便成为节点 r 的输入．若输入 w^Tx 为正，则节点 r 输出 1，若输入 w^Tx 为负，则节点 r 输出 0．也就是说，把节点 r 的输出用 z 表示的话，z 由式 (6.2) 所示的阈函数 G_T

$$z=G_T(w^Tx) \tag{6.8}$$

确定．

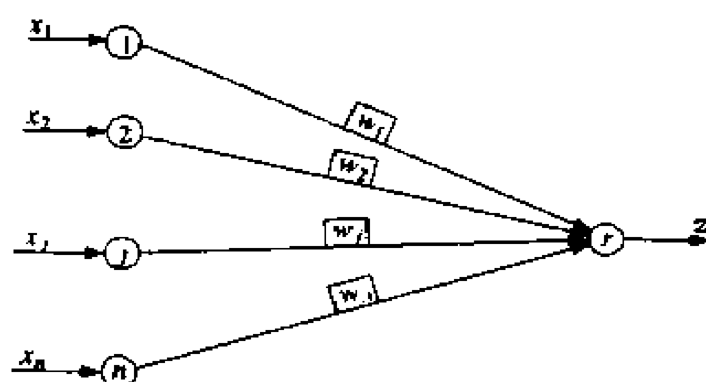


图 6.5　基本感知器

要把这种基本感知器用于 6.2.1 的模式识别必须设定使输入为 $q^1,q^2,\ldots,q^{N'}$ 时输出 $z=1$，输入为 $q^{N'+1},q^{N'+2},\ldots,q^N$ 时输出 $z=0$ 的权系数 w．但是，这种权系数 w 都是一次不等式 (6.6) 或不等式 (6.7) 的解，因此，只要找到这些不等式的解，就做成了达到所期待目的模式识别机器．

为求出满足式 (6.6) 的权系数 w，考虑下述被称为**改错学习** (learning by error correction) 的方法．首先，在适当设定了权系数的感知器里，逐次输入待识别的模式，等到出现了错误输出时再修改权系数．经过这种手续后，可以期待最终得出对任何模式都有正确输出的权系数．概括该算法如下．

算法 PERCEPTRON (感知器的学习)

第一步 (初始化): 确定权系数的适当初始解 $w^{(0)} \in R^n$ 及正的常数 η, 令 $k := 0$.

第二步 (计算输出): 令 [1] $s := k(\text{mod}N) + 1$. 计算针对模式向量 $q^s \in R^n$ 的输出 $z = G_T\left((w^{(k)})^T q^s\right)$. 若连续 N 次都得到正确输出, 则停止.

第三步 (修正权系数): 若 $1 < s < N'$, 则

$$w^{(k+1)} := \begin{cases} w^{(k)}, & \text{当} z = 1 \text{ (正确输出) 时} \\ w^{(k)} + \eta q^s, & \text{当} z = 0 \text{ (错误输出) 时} \end{cases} \tag{6.9}$$

若 $N' + 1 \le s \le N$, 则

$$w^{(k+1)} := \begin{cases} w^{(k)} - \eta q^s, & \text{当} z = 1 \text{ (错误输出) 时} \\ w^{(k)}, & \text{当} z = 0 \text{ (正确输出) 时} \end{cases} \tag{6.10}$$

令 $k := k + 1$, 回到第二步.

下面简单说明该算法第三步所执行的修正权系数的想法. 当输入模式向量 q^s $(1 \le s \le N')$ 时, 若网络输出正确信号 $z = 1$, 则不修正权系数的值. 但是, 若网络错误输出 $z = 0$, 就要强化与模式向量的第 j 分量 q_j^s 为正的节点 j 的接合. 反之, 要弱化与 $q_j^s < 0$ 的节点 j 的接合 (参照式 (6.9)). 也就是, 要按使对模式向量 q^s 的输出成为 1 的方向修正权系数. 对模式向量 q^s $(N' + 1 \le s \le N)$ 的权系数的修正也是基于同样的想法.

关于感知器的学习算法有下面的定理成立.

定理 6.1 (感知器的收敛定理): 如果模式向量 $q^1, q^2, \ldots, q^N$ 线性可分, 则对任意的初始解 $w^{(0)} \in R^n$ 及 $\eta > 0$, 感知器的学

1) 下面的 $k(\text{mod}\,N)$ 表示 k 除 N 的余数.

习算法 PERCEPTRON 一定在有限次迭代后停止．另外，若采用最后的迭代所得到的权系数，则感知器能正确识别所有的模式向量 $q^1, q^2, \ldots, q^N$

在此省略严格的证明，可以如下直观理解该定理是正确的．首先，满足不等式 (6.6) 的权系数 w 的集合，在 w 的空间里面，是以 N 个半空间 $\{w|(q^s)^T w > 0\}$，$(s = 1, 2, \ldots, N')$ 以及 $\{w|(q^s)^T w < 0\}$ $(s = N'+1, N'+2, \ldots, N)$ 的共同部分所表示的领域．特别是，当模式向量 $q^1, q^2, \ldots, q^N$ 线性可分时，该领域非空．图 6.6 示意了向量 w 的维数为 2，模式数 $N = 4$ 及 $N' = 2$ 的情况．这时，不等式 (6.6) 的解集合是阴影部分．（注意对应 q^1, q^2 的半空间和对应 q^3, q^4 的半空间的方向相反．）上述学习算法 PERCEPTRON 的各次迭代里，当前的 $w^{(k)}$ 若被包含在对应模式向量 q^s 的半空间里则不动，若不被包含则向着该半空间垂直移动，其步长为 $\eta > 0$．可以看出，只要解集合非空，按图 6.6 所示重复这种移动后总能在一定时候到达解集合里．

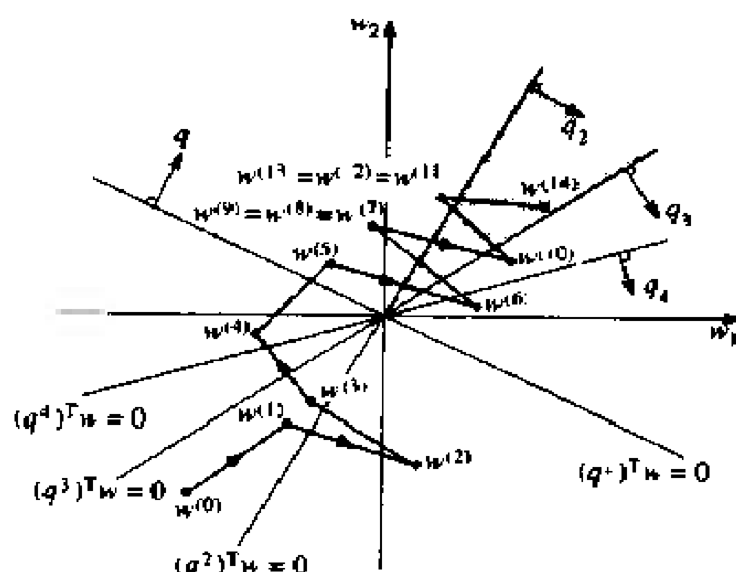


图 6.6　在权系数的空间里学习算法的运行这里，解集合（阴影部分）是 $\{w\ (q^1)^T w > 0, (q^2)^T w > 0, (q^3)^T w < 0, (q^4)^T w < 0\}$

感知器算法是非常简单易懂的方法，但是，使用固定步长有可能引起到达解集合所要的迭代次数不必要地多. 因此，在算法的各次迭代里，当前的 $w^{(k)}$ 不包含在对应模式向量 q^s 的半空间里的时候，可以考虑根据点 $w^{(k)}$ 到该半空间的距离决定实际移动步长. 直观说来，这相当于在对某个模式有错误输出时，按该差错程度决定权系数的修正量.

算法上，该操作相当于针对联立一次不等式使用被称为**逐次投影法** (successive projection method) 或 **松弛法** (relaxation method) 的方法. 为进行正确的数学处理，在此考虑求满足不等式 (6.7) 的 w. 现在假定点 $w^{(k)}$ 不包含在半空间 $\{w \mid (q^s)^T w > 1\}$ 里，也就是说 $(q^s)^T w^{(k)} < 1$，则可以说点 $w^{(k)}$ 在该半空间的正投影点为 $w^{(k)} + (1-(q^s)^T w^{(k)})q^s/\|q^s\|^2$ (图 6.7).

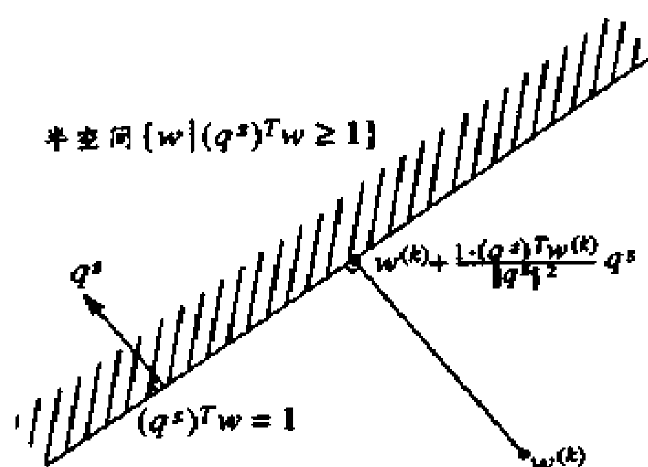


图 6.7　至半空间 $(q^s)^T w \geq 1$ 上的投影

选取这样得到的点为 $w^{(k+1)}$ 的方法便是逐次投影法. 再者，有很多情况下，不用正投影点本身作为 $w^{(k+1)}$，而用稍微大一点的步长确定 $w^{(k+1)}$ 的话，从收敛速度这一点来说要好. 用这种方法，在学习算法第三步里，代替 (6.9)，(6.10)，若 $1 \leq s \leq N'$，

则以

$$w^{(k+1)} = \begin{cases} w^{(k)}, & z=1 \text{ 时} \\ w^{(k)} + \lambda \dfrac{1-(q^s)^T w^{(k)}}{\|q^s\|^2} q^s, & z=0 \text{ 时} \end{cases} \tag{6.11}$$

若 $N'+1 \leq s \leq N$, 则以

$$w^{(k+1)} = \begin{cases} w^{(k)} - \lambda \dfrac{1+(q^s)^T w^{(k)}}{\|q^s\|^2} q^s, & z=1 \text{ 时} \\ w^{(k)}, & z=0 \text{ 时} \end{cases} \tag{6.12}$$

来更新权系数 $w^{(k)}$. 这里，式 (6.11)、(6.12) 中的 λ 是调整步长的常数. 现在已经知道，若确定满足 $0 < \lambda < 2$ 的 λ, 所生成的权系数列 $\{w^{(k)}\}$ 收敛于不等式 (6.7) 的解集合. 因此，在这种情况下，满足狭义不等式 (6.6) 权系数 w 也可经有限次迭代得到. 再者，正如上面所述，实际计算中经常选取大一点的步长，$1 < \lambda < 2$. 这时该方法成为对联立一次不等式的**逐次过缓法** (successive over-relaxation method, SOR 法). 这样，确定感知器权系数的问题，因为本质上仅是解不等式 (6.6) 或 (6.7), 把针对联立一次不等式的各种方法纳入学习算法是可能的.

6.3 多层网络的学习

图 6.5 的基本感知器可看成以节点 $1,2,\ldots,n$ 为输入层，节点 r 为输出层的双层网络. 因结构单纯，不适于非线性可分模式集合的识别等等，其能力有限. 为改善这类感知器的缺点，采用**多层网络** (multilayer network) 的尝试活跃起来了.

为简单起见，在此考虑图 6.8 所示的三层网络. 第一层 (输入层) 的各节点 $j=1,2,\ldots,n$ 接收输入信号 x_j 并把它传递到第二层的节点 $i=1,2,\ldots,m$. 这时，若节点 j 和节点 i 的权系数为 v_{ij} 的话，则到第二层的节点 i 的输入的总和成为 $\sum_{j=1}^{n} v_{ij} x_j$.

节点 i 运用适当的输出函数 G, 把该输入变换成

$$y_i = G\left(\sum_{j=1}^{n} v_{ij} x_j\right) \tag{6.13}$$

后送至第三层 (输出层) 的节点 r. 若第二层的节点 i 和节点 r 的权系数为 w_i, 则送到节点 r 的输入成为 $\sum_{i=1}^{m} w_i y_i$. 节点 r 变换成

$$z = G\left(\sum_{i=1}^{m} w_i y_i\right) \tag{6.14}$$

后成为网络的输出. 这样, 该网络经对输入 $x_1, \ldots, x_n$ 作在第二层和第三层两阶段的变换后, 最终输出信号 z. 这时, 第二层因从网络的外部看不见, 有时也称为隐层 (hidden layer).

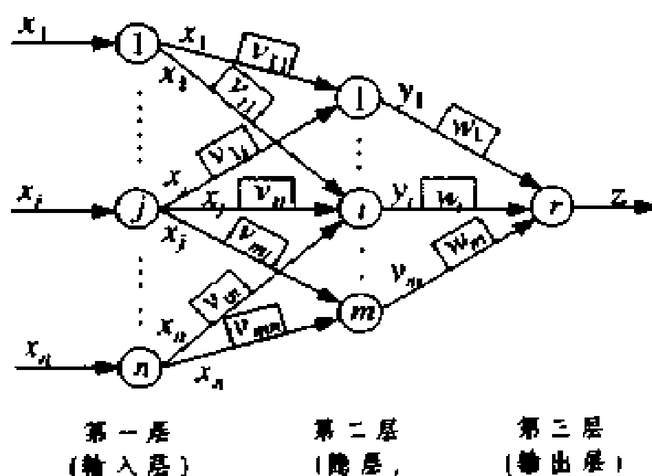


图 6.8 多层网络

现在把待识别的 N 个模式向量记为 $x^s = (x_1^s, x_2^s, \ldots, x_n^s)\ (s = 1, 2, \ldots, N)$, 各自分别对应的由网络给出的希望输出设为 $d^s\ (s = 1, 2, \ldots, N)$. 在此, 与前节限定输出为 0, 1 的二值模型不同, 本节所考虑的各节点的输出可取 0 和 1 之间的连续值. 另外, 像后面要说的那样, 在以 S- 型函数 G_S 为输出函数的情况下,

输出取开区间 (0,1) 内的值而不允许取正好为 0 或 1 的值. 于是, 在这种情况下, 选取 (比如说) 0.1 和 0.9 作为希望的输出 $d^s\ (s = 1, 2, \ldots, N)$.

网络的权系数为 $v = (v_{11}, v_{12}, \ldots, v_{mn})$, $w = (w_1, w_2, \ldots, w_m)$ 时, 对应各模式向量 x^s 的第二层的输出记为 $y^s(v) = (y_1^s(v), y_2^s(v), \ldots, y_m^s(v))$, 接受该输出后第三层 (节点 r) 给出网络的最终输出记为 $z^s(v, w)$. 这时, 目标成为确定对所有的输入模式, 网络给出正确输出的权系数的值, 也就是, 解下面的以权系数 (v, w) 为变量的联立非线性方程组

$$\begin{cases} z^1(v,w) & = & d^1 \\ z^2(v,w) & = & d^2 \\ & \vdots & \\ z^N(v,w) & = & d^N \end{cases} \tag{6.15}$$

像上面所说过的那样, 以下考虑的第二层的各节点 i 的输出 y_i 和第三层的节点 r 的输出 z 是 0 和 1 之间的实数, 并采用式 (6.3) 的 S- 型函数 G_S 作为各节点的输出函数. 这时, 用下式所定义的 $y_i^s(v)$,

$$y_i^s(v) = G_S\left(\sum_{j=1}^{n} v_{ij} x_j^s\right) \quad (i = 1, 2, \ldots, m,\ \ s = 1, 2, \ldots, N) \tag{6.16}$$

把函数 $z^s(v, w)$ 定义为

$$z^s(v,w) = G_S\left(\sum_{i=1}^{m} w_i y_i^s(v)\right) \quad (s = 1, 2, \ldots, N) \tag{6.17}$$

联立方程式组 (6.15) 所含的变量和方程式的数目分别是 $nm + m$ 和 N, 故该方程组一般无解. 因此, 为构造使识别误差尽量小

的网络，考虑使对应输入模式 x^s 的正确输出和实际输出的二乘误差

$$e^s(v,w) = \frac{1}{2}(z^s(v,w) - d^s)^2 \tag{6.18}$$

的和最小化的如下问题.

$$\text{目标函数:}\quad E(v,w) \equiv \sum_{s=1}^{N} e^s(v,w) \quad \rightarrow \quad \text{最小} \tag{6.19}$$

根据式 (6.16), (6.17) , (6.18), 因函数 e^s 是可微的 S- 型函数 G_S 的合成函数，关于变量 (v,w) 可微. 下面实际计算一下函数 e^s 关于各变量的偏微分.

首先，根据式 (6.18) 和 (6.17), 关于变量 w_i 的偏微分是

$$\begin{aligned}\frac{\partial e^s(v,w)}{\partial w_i} &= (z^s(v,w) - d^s)\frac{\partial z^s(v,w)}{\partial w_i} \\ &= (z^s(v,w) - d^s) G'_S\left(\sum_{i=1}^{m} w_i y_i^s(v)\right) y_i^s(v)\end{aligned} \tag{6.20}$$

这里， G'_S 是 S- 型函数 G_S 的导函数，由式 (6.4) 给出.

其次，考虑函数 e^s 关于变量 v_{ij} 的偏微分. 首先，根据式 (6 18) 和 (6.17) 有

$$\begin{aligned}\frac{\partial e^s(v,w)}{\partial v_{ij}} &= (z^s(v,w) - d^s)\frac{\partial z^s(v,w)}{\partial v_{ij}} \\ &= (z^s(v,w) - d^s) G'_S\left(\sum_{i=1}^{m} w_i y_i^s(v)\right) \sum_{i=1}^{m} w_i \frac{\partial y_i^s(v)}{\partial v_{ij}}\end{aligned} \tag{6.21}$$

进一步，根据式 (6.16) 有

$$\frac{\partial y_i^s(v)}{\partial v_{ij}} = G'_S\left(\sum_{j=1}^{n} v_{ij} x_j^s\right) x_j^s \tag{6.22}$$

结果，由式 (6.21), (6.22), 函数 e^s 关于变量 v_{ij} 的偏微分可表示为

$$\frac{\partial e^s(v,w)}{\partial v_{ij}} = (z^s(v,w) - d^s) G'_S \cdot \left(\sum_{i=1}^{m} w_i y_i^s(v)\right) \sum_{i=1}^{m} w_i G'_S \left(\sum_{j=1}^{n} v_{ij} x_j^s\right) x_j^s \quad (6.23)$$

利用神经网络计算函数值 $e^s(v,w)$ 的时候，先定下权系数的值 (v,w)，第一层里输入模式向量 $x^{(s)}$. 把其数据送往第二层，变换成式 (6.16) 的向量 $y^{(s)}(v)$ 后再送往第三层. 在第三层里进行式 (6.17) 的变换，因要输出信号 $z^{(s)}(v,w)$，把它与对应于输入模式 $x^{(s)}$ 的希望输出 $d^{(s)}$ 比较后，就可求出式 (6.18) 的误差函数值 $e^s(v,w)$.

再次考虑计算函数 e^s 的微分的情况. 权系数的值为 (v,w)，对应输入模式 $x^{(s)}$ 得到输出 $z^{(s)}(v,w)$ 时，函数 e^s 的关于 w_i 偏微分 $\partial e^s(v,w)/\partial w_i$ 由式 (6.20) 给出，这个值是把输出的误差信号 $z^{(s)}(v,w) - d^{(s)}$ 和输出函数的微分 $G'_S(\sum_{i=1}^{m} w_i y_i^s(v))$ 从第三层的节点 r 送回到第二层的各节点 i, 可在第二层计算 [1]. 进一步，根据式 (6.23)，若知道 $(z^s(v,w) - d^s)G'_S(\sum_{i=1}^{m} w_i y_i^s(v))$ 和 $w_i G'_S\left(\sum_{j=1}^{n} v_{ij} x_j^s\right)$ $(i = 1, 2, \cdots, m)$ 的值，就可计算出 $\partial e^s(v,w)/\partial v_{ij}$. 前者也就是在第二层里为计算式 (6.20) 的 $\partial e^s(v,w)/\partial w_i$ 由第三层的节点 r 送到第二层的节点 i 的信息，后者因是在第二层的各节点 i 处已经可以求出的量 [2]，它们都是可以从第二层的各节点传到第一层节点 j 的信息.

这样，在第三层观测到的关于误差的信息，与输入信号反向

1) 对 S- 型函数 G_S 来说，因成立式 (6.4)，微分 $G'_S\left(\sum_{i=1}^{m} w_i y_i^s(v)\right)$ 的值可从实际输出值 $G_S\left(\sum_{i=1}^{m} w_i y_i^s(v)\right)$ 直接求出

2) 参照前脚注.

传递，在第二层的节点 $i\ (i=1,2,\ldots,m)$ 可以计算出 $\partial e^s(v,w)/\partial w_i$，在第一层的节点 j 可以计算出 $\partial e^s(v,w)/\partial v_{ij}$．基于该原理求误差函数的微分称为**误差逆传播** (error back propagation)．

结果，计算问题 (6.19) 的目标函数的梯度向量

$$\nabla E(v,w)=\sum_{s=1}^{N}\nabla e^s(v,w) \tag{6.24}$$

时，先固定权系数 (v,w) 的值、依次把模式向量 $x^1,\ldots,x^N$ 输入网络，对它们分别用误差逆传播法求出 $\nabla e^1(v,w),\nabla e^2(v,w),\ldots,\nabla e^N(v,w)$ 后取和便可．

因而，对问题 (6.19)，若用**最速下降法** (参照 4.2 节)，可得到如下的计算步骤．首先，选取权系数的适当的初始值 $(v^{(0)},w^{(0)})$ 在第 k 次迭代里，固定权系数为现在的值 $(v^{(k)},w^{(k)})$，把 N 个模式向量输入网络，根据式 (6.24) 求出 $\nabla E(v^{(k)},w^{(k)})$．其次，用适当的步长 $t^{(k)}>0$ 根据下式

$$v_{ij}^{(k+1)}:=v_{ij}^{(k)}-t^{(k)}\frac{\partial E(v^{(k)},w^{(k)})}{\partial v_{ij}} \quad (i=1,2,\ldots,m,\ j=1,2,\ldots,n) \tag{6.25}$$

$$w_i^{(k+1)}:=w_i^{(k)}-t^{(k)}\frac{\partial E(v^{(k)},w^{(k)})}{\partial w_i} \quad (i=1,2,\ldots,m) \tag{6.26}$$

来更新权系数的值，进入下一次迭代．步长可对目标函数 $E(v,w)$ 用一维搜索确定，通常也采用预先决定适当的固定步长，在所有的迭代里都用这个值的简便方法．在适当的条件下可断言，这种最速下降法所得到的权系数列 $\{(v^{(k)},w^{(k)})\}$ 收敛于二乘误差的最小化问题 (6.19) 的局部最优解．

最初提出误差逆传播法的 Rumelhart、Hinton、Williams 并不是把权系数固定为 $(v^{(k)},w^{(k)})$ 后，输入所有的 N 个模式向量来求出 $\nabla E(v^{(k)},w^{(k)})$．他们给出的是，输入某一个模式向

量 x^s，利用此时得到的（对于模式向量 x^s 的）误差函数的梯度 $\nabla e^s(v^{(k)}, w^{(k)})$（参照式 (6.21)，(6.24)），由

$$v_{ij}^{(k+1)} := v_{ij}^{(k)} - t^{(k)} \frac{\partial e^s(v^{(k)}, w^{(k)})}{\partial v_{ij}}$$
$$(i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n) \qquad (6.27)$$
$$w_i^{(k+1)} := w_i^{(k)} - t^{(k)} \frac{\partial e^s(v^{(k)}, w^{(k)})}{\partial w_i} \quad (i = 1, 2, \ldots, m) \quad (6.28)$$

直接给出更新权系数的方法．另外 Rumelhart 等认为，只要选定步长 $t^{(k)}$ 为足够小的常数，每输入一个模式向量更新权系数的方法 (6.27) (6.28) 可以看作是近似执行针对问题 (6.19) 的最速下降法 (6.25)–(6.26)，因而主张该方法是合适的．但是，模式向量的数目多起来时，一次输入所有模式向量的间隔里、权系数的值实际上应该变化相当大，上述 Rumelhart 等的解释未必妥当．

改变一下看法，式 (6.27)–(6.28) 的方法可以看成是对联立方程组 (6.15) 推广前节的逐次投影法的结果．为记号上的方便，以下令 $u = (v, w)$．现设权系数的值为 $u^{(k)} = (v^{(k)}, w^{(k)})$，把模式向量 x^s 输入网络．如果 $z^s(u^{(k)}) \neq d^s$，则权系数向量 u 的空间里点 $u^{(k)}$ 不在曲面 $\{u \mid z^s(u) = d^s\}$ 上（参照图 6.9）．因此，在点 $u^{(k)}$ 对方程式 $z^s(u) = d^s$ 进行线性近似成为

$$z^s(u^{(k)}) + \nabla z^s(u^{(k)})^T (u - u^{(k)}) = d^s \qquad (6.29)$$

求出点 $u^{(k)}$ 在由式 (6.29) 所定的超平面上的投影点，并记之为 $u^{(k+1)}$．点 $u^{(k+1)}$ 的值能容易求出，

$$u^{(k+1)} := u^{(k)} - t^{(k)} \nabla z^s(u^{(k)}) \qquad (6.30)$$

这里，步长 $t^{(k)}$ 由

$$t^{(k)} := \frac{z^s(u^{(k)}) - d^s}{\|\nabla z^s(u^{(k)})\|^2} \tag{6.31}$$

给出[1]. 因此，根据式 (6.18) 有

$$\nabla e^s(v, w) = (z^s(v, w) - d^s)\nabla z^s(v, w)$$

现在，$u = (v, w)$, $\nabla z^s(u^{(k)}) = (\nabla_v z^s(v^{(k)}, w^{(k)}), \nabla_w z^s(v^{(k)}, w^{(k)}))$, 因而步长 $t^{(k)}$ 的值暂且不管时，式 (6.30) 与式 (6.27) (6.28) 同形. 对于在每次迭代中选取一个模式向量 x^s, 按式 (6.30), (6.31) 更新权系数的逐次投影法，在联立方程组 (6.15) 有解的时候，可以断言，如果选取初始解 $u^{(0)} = (v^{(0)}, w^{(0)})$ 与该解充分接近的话，所生成的点列 $\{(v^{(k)}, w^{(k)})\}$ 收敛于解 (局部收敛性). 但是，能否得到与初始解的选法无关的解的收敛性 (全域收敛性) 以及联立方程组 (6.15) 无解时的收敛性等等？这些问题的解答还不是很清楚.

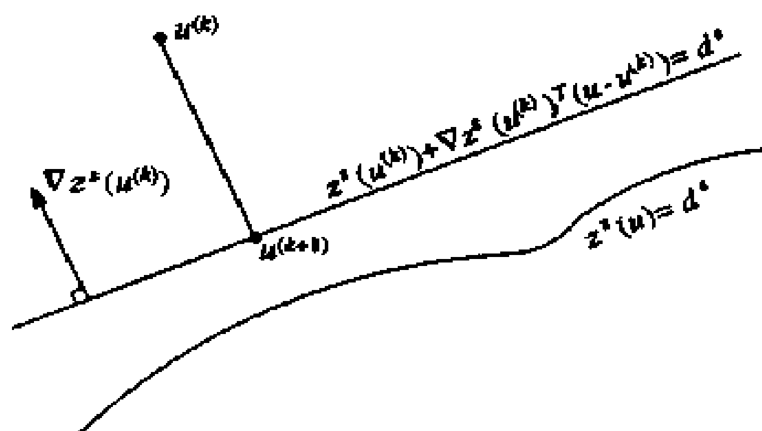


图 6.9 至近似超平面上的投影

1) 与前节的逐次投影法以不等式为对象相比，这儿的方法以方程式 (等式) 为对象，因此根据目前的在 $u^{(k)}$ 处 $z^s(u^{(k)}) - d^s$ 的值步长可能为正也可能为负. 另外，和前节方法一样，也可在式 (6.31) 的右边乘以常数 $0 < \lambda < 2$ 作为步长 $t^{(k)}$ (参照式 (6.10), (6.11)).

6.4 Hopfield 网络和组合优化

考虑如下 0-1 非线性 (二次) 规划问题.

$$\begin{array}{ll} \text{目标函数:} & E(x) = -\frac{1}{2}x^T W x + \theta^T x \longrightarrow \text{最小} \\ \text{约束条件:} & x_i = 0,1 \ (i = 1,2,...,n) \end{array} \tag{6.32}$$

这里, 变量为 n 维向量 $x = (x_1, x_2, ..., x_n)^T$, $W = (w_{ij})$ 和 $\theta = (\theta_1, \theta_2, ..., \theta_n)^T$ 分别为 $n \times n$ 常数矩阵和 n 维常数向量. 进一步, 矩阵 W 为对称的, 即假定

$$w_{ij} = w_{ji} \quad (i,j = 1,2,...,n \ (i \neq j)) \tag{6.33}$$

而且对角分量

$$w_{ii} = 0 \quad (i = 1,2,...,n) \tag{6.34}$$

与问题 (6.32) 相关连, 考虑有 n 个节点 $i = 1,2,...,n$, 节点 j 到节点 i 的权系数为 w_{ij}, 节点 i 的阀值为 θ_i 的神经网络 (参照图 6.1 和图 6.2). 该网络和前节所考察过的多层网络不同, 尽管它也是所有的节点间均存在结合的相互结合型的网络, 但从节点 j 到节点 i 的权系数 w_{ij} 等于反方向的权系数 w_{ji} (对称性), 以及各节点不存在自身反馈 (式 (6.34)). 这种网络一般称为 **Hopfield 网络**, 函数 E 称为 **能量函数** (energy function).

这里, 固定 Hopfield 网络的所有权系数 $W = (w_{ij})$ 和阀值 θ 为预先取定的值, 考虑在各节点的输出函数 G 取为式 (6.2) 的阀函数 G_T 时网络的情况. 这儿设各节点的输出为 0 或 1, 在各时刻选取一个节点进行输入输出变换. (假定时间为离散的, 每个节点都在某个时间内被选出一回.) 现在, 在时刻 k 设各

节点的输出为 $x_i^{(k)}$ $(i=1,2,\dots,n)$，若把进行输入输出变换的节点记为 q，则在时刻 $k+1$ 的各节点的输出由

$$x_q^{(k+1)} := \begin{cases} 1, & 在 \ \sum_{j=1}^{n} w_{qj}x_j^{(k)} > \theta_q \ 时 \\ x_q^{(k)}, & 在 \ \sum_{j=1}^{n} w_{qj}x_j^{(k)} = \theta_q \ 时 \\ 0, & 在 \ \sum_{j=1}^{n} w_{qj}x_j^{(k)} < \theta_q \ 时 \end{cases} \tag{6.35}$$

$$x_i^{(k+1)} = x_i^{(k)} \qquad (i=1,2,\dots,n \quad i \neq q) \tag{6.36}$$

给出[1]．这时，考虑问题 (6.32) 的目标函数 E 的变化量，根据假定 (6.33)，(6.34) 和式 (6.36) 有

$$E(x^{(k+1)}) - E(x^{(k)}) = -(x_q^{(k+1)} - x_q^{(k)})\left(\sum_{j=1}^{n} w_{qj}x_j^{(k)} - \theta_q\right) \tag{6.37}$$

但是，根据式 (6.35)，仅当 $x_q^{(k)}=0$ 而且 $\sum_{j=1}^{n} w_{qj}x_j^{(k)} > \theta_q$ 时，或者 $x_q^{(k)}=1$ 而且 $\sum_{j=1}^{n} w_{qj}x_j^{(k)} < \theta_q$ 时，成立 $x_q^{(k+1)} \neq x_q^{(k)}$．在两种情况下式 (6.37) 的右边都为负，故成立 $E(x^{(k+1)}) < E(x^{(k)})$．另外，$x_q^{(k+1)} = x_q^{(k)}$ 时，显然 $E(x^{(k+1)}) = E(x^{(k)})$，结果，可以知道目标函数值的数列 $\{E(x^{(k)})\}$ 为非增的．

以上分析意味着依据式 (6.35) 和 (6.36) 的迭代成为对问题 (6.32) 的一种下降法．特别是，当点 $x^{(k)}$ 的值变化时目标函数值也一定严格减少，而且点 $x^{(k)}$ 可取的值只有有限 (2^n) 种，因而该迭代在有限时间内进入某个定常状态，可以断定在此之后 $x^{(k)}$ 的值无变化．进一步，变动达到这种定常状态后的 $x^{(k)}$ 的任意一个分量变化后，目标函数值都不会减少，因此可看成问

1) 式 (6.36) 表示 $x_q^{(k+1)} = G_T(\sum_{j=1}^{n} w_{qj}x_j^{(k)} - \theta_q)$．这里，$\sum_{j=1}^{n} w_{qj}x_j^{(k)} = \theta_q$ 的时候，设输出 $x_q^{(k+1)}$ 仍然为该节点现在的输出 $x_q^{(k)}$．

题 (6.32) 的一个局部最优解 [1].

因问题 (6.32) 存在有多个局部最优解，哪个局部最优解能成为上述 Hopfield 网络的定常状态，在很大程度上取决于从哪个初始解出发. 这是整个非线性最优化下降法所共有的性质，用下降法求全局最优解是非常困难的. 因此实际上在很多时候采取对各种初始解进行计算，从所得到的几个局部最优解中选取最好者. 此外，还有这么一种想法. 放松在各迭代里目标函数值单调减少的条件，根据情况允许函数值增加，摆脱不满意的局部最优解后到达更好的解. 在 Hopfield 网络里，通过引进如下的概率机制，这种想法是可行的.

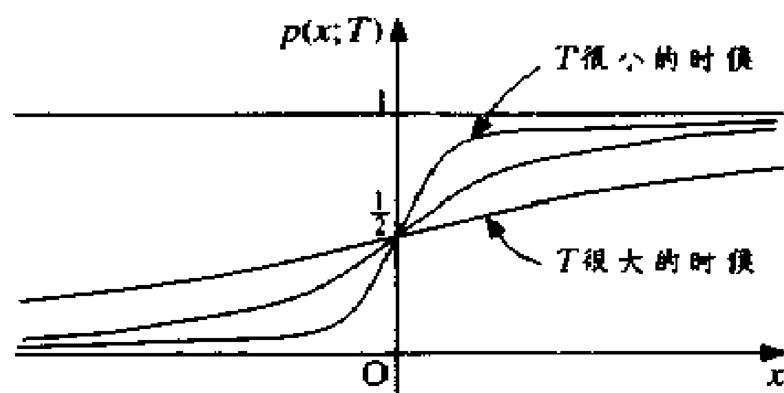


图 6.10 S- 型函数 $p(x,T)=1/(1+\exp(-x/T))$

首先，定义包含参数 $T>0$ 的 S- 型函数

$$p(x,T)=\frac{1}{1+\exp(-x/T)} \tag{6.38}$$

如图 6.10 所示，这个函数随着 T 变小，在 $x=0$ 附近变动很大，$T\to 0$ 时接近阀函数. 现在，假定在时刻 k 网络在各节点的输出为 $x^{(k)}$，该时刻变换输入输出的节点和以前一样设为 q，

1) 给定双值向量 $x=(x_1,x_2,\cdots,x_n)$ 和 $y=(y_1,y_2,\cdots,y_n)$，成立 $x_i\neq y_i$ 的分量的个数称为 x 和 y 的汉明距离. 以定常状态 $x^{(k)}$ 点为中心，用汉明距离为测度的半径为 1 的球里面，函数 E 达最小. 在这个意义下，成为局部最优解.

则到达节点 q 的输入为 $\sum_{j=1}^{n} w_{qj} x_j^{(k)}$. 因此，用函数 $p(x;T)$ 可给定在时刻 $k+1$ 节点 q 输出 1 的概率为

$$\mathrm{Prob}\,(x_q^{(k+1)}=1)=p\left(\sum_{j=1}^{n} w_{qj} x_j^{(k)}-\theta_q;T\right) \tag{6.39}$$

输出为 0 的概率为 $\mathrm{Prob}\,(x_q^{(k+1)}=0)=1-\mathrm{Prob}\,(x_q^{(k+1)}=1)$. 也就是，与式 (6.35) 中根据输入值 $\sum_{j=1}^{n} w_{qj} x_j^{(k)}$ 和阀值 θ_q 的大小关系确切给定下一时刻的输出不同，该方法把输入值与阀值相比，大得越多则下一时刻输出为 1 的概率就越接近于 1, 但仍然留有输出 0 的可能性. 相反，输入值比阀值小的时候也可能有输出 1 的情况. 其结果，函数值 $E(x^{(k)})$ 并不限于单调减少，在有些迭代里反而增加. 这种采用了概率机制的神经网络被称为**波尔兹曼机** (Boltzmann machine).

在波尔兹曼机里，称参数 T 为网络的温度. 根据函数 $p(x;T)$ 的性质 (参照图 6.10), 可以看出，温度 T 越高，产生随机举动的倾向就越大，能量函数值增加的概率就越大，T 接近于 0 的时候，其行为就接近于式 (6.35) 的确定性的下降法. 也就是，可以认为温度高的时候，网络的输出变化很活，温度低的时候，容易落在稳定的定常状态. 因此，有人提出了采纳一种叫**模拟退火**(simulated annealing) 思想的方法，在初始阶段把网络的温度设得高点，逃离收敛到不满意的局部最优解的情况. 随着计算进行下去，慢慢地把温度降下，让它落进满意的解.

上述方法里各变量 x_i 的值被限定取 0, 1 或 2, 放松该条件，可取 0 和 1 之间的连续值时候，问题 (6.32) 也是可解的. 为研究这种情况，考虑如下的二次规划问题.

$$\begin{aligned} &\text{目标函数:} \quad E(x)=-\frac{1}{2}x^T W x+\theta^T x \longrightarrow \text{最小} \\ &\text{约束条件:} \quad 0 \le x_i \le 1 \;\; (i=1,2,\ldots,n) \end{aligned} \tag{6.40}$$

连续变量问题 (6.40) 的可行域是空间 R^n 内的立方体，各顶点对应于 0-1 变量问题 (6.32) 的可行解．设 x^* 为问题 (6.40) 的任意局部最优解，根据最优性一次必要条件 (定理 4. 8),

$$\begin{cases} x_i^* = 0 & \Longrightarrow \dfrac{\partial E(x^*)}{\partial x_i} \geq 0 \\ 0 < x_i^* < 1 & \Longrightarrow \dfrac{\partial E(x^*)}{\partial x_i} = 0 \\ x_i^* = 1 & \Longrightarrow \dfrac{\partial E(x^*)}{\partial x_i} \leq 0 \end{cases} \tag{6.41}$$

进一步根据最优性的二次必要条件 (定理 4. 9), 矩阵

$$\left[\frac{\partial^2 E(x^*)}{\partial x_i \partial x_j};\ i, j \in J\right] \tag{6.42}$$

必须为半正定的．这里，J 为满足 $0 < x_i^* < 1$ 的 i 集合．但是式 (6.42) 的矩阵正好是从 W 里取出对应下标集合 J 的行和列后得到的子矩阵 $W_{JJ} = (w_{ij}\,;\, i, j \in J)$ 乘 -1 后的矩阵，另外根据式 (6.34) 有 $\operatorname{trace} W_{JJ} = 0$, 因此式 (6.42) 的矩阵一般有正和负的特征值[1]．半正定值矩阵的特征值全部非负，因此 $J \neq \emptyset$ 的时候，也就是 x^* 包含 0, 1 以外的分量时，式 (6.42) 一般不成立．据此，问题 (6.40) 的所有局部最优解为可行域 (立方体) 顶点，可以认为问题 (6.40) 和问题 (6.32) 等价．

根据以上讨论，我们知道代替问题 (6.32) 也可以考虑问题 (6.42)．其次，在开区间 (0,1) 上定义的凸函数 h, 使得当 $\xi \to 0$ 或者 $\xi \to 1$ 的时候，成立 $h(\xi) \to +\infty$ (图 6.11)．利用函数 h, 定义问题

---

1) $n \times n$ 矩阵 A 的对角分量的和 $\sum_{i=1}^{n} a_{ii}$ 称为 A 的迹，用 $\operatorname{trace} A$ 表示．若矩阵 A 的特征值为 $\lambda_1, \lambda_2, \cdots, \lambda_n$ 的话，则 $\operatorname{trace} A = \sum_{i=1}^{n} \lambda_i$ 成立．因此，如果 $\operatorname{trace} A = 0$, 只要所有特征值并不都为 0, 就一定存在有正和负的特征值．

$$
\begin{aligned}
&\text{目标函数:} \quad \hat{E}(x) = -\frac{1}{2}x^T W x + \theta^T x + \sum_{i=1}^{n} \frac{h(x_i)}{R_i} \longrightarrow \text{最小} \\
&\text{约束条件:} \quad 0 < x_i < 1 \quad (i = 1, 2, \dots, n)
\end{aligned} \tag{6.43}
$$

这里，R_i 为正参数．从图 6.11 可看出，函数 $\hat{E}$ 是一种罚函数（参照 4.7 节）[1]．参数 R_i 的值选得足够大时，问题 (6.43) 的最优解成为问题 (6.40) 的最优解的好近似解．另外，函数 $\hat{E}$ 的定义域为开集合 $\{x \in R^n \mid 0 < x_i < 1, (i = 1, 2, \dots, n)\}$，在其局部最优解处成立 $\nabla \hat{E}(x) = 0$，因此可把问题 (6.43) 看成是实质上无约束的问题．于是，对该问题，可以运用第 4 章所述的无约束最优化方法．

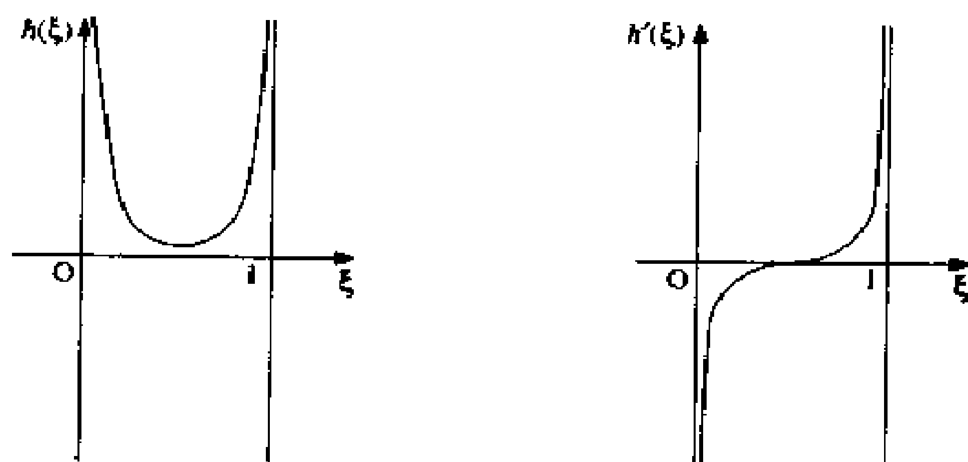


图 6.11 函数 $h(\xi)$ 及其导函数 $h'(\xi)$

与此同时，在把变量 x 看成时间参数 t 的函数的同时，导入新变量 u，也可以考虑下面的微分方程组．这儿把时间考虑成连续的．

---

1) 在惩罚违反约束条件的意义下，通常的罚函数在可行域外部的点取很大的值．而函数 h 则是在可行域的边界处设置障碍，使函数 h 在最小化时候产生的点列出不了可行域．因此，类似 $\hat{E}$ 的函数有时候也称为**障碍函数** (barrier function) 或**内点罚函数** (interior penalty function)

$$C_i \frac{du_i}{dt} = -\sum_{j=1}^{n} w_{ij} x_j - \theta_i - \frac{u_i}{R_i},$$

$$u_i = h'(x_i) \qquad (i = 1, 2, \ldots, n) \tag{6.44}$$

这里，C_i 为正常数，$h'(\cdot)$ 为凸函数 h 的导函数. 从图 6.11 可看出，$h'(\cdot)$ 呈 S- 型函数的反函数的形状，注意其导函数 (函数 h 的二次导函数) 处处成立有 $h''(\xi) > 0$. 对应于适当的初始解，微分方程组 (6.44) 的解记为 $(x(t), u(t))$, 则有

$$\begin{aligned}
\frac{d\hat{E}(x(t))}{dt} &= \nabla \hat{E}(x(t))^T \frac{dx(t)}{dt} \\
&= \sum_{i=1}^{n} \left(\sum_{j=1}^{n} w_{ij} x_j(t) + \theta_i + \frac{u_i(t)}{R_i} \right) \frac{dx_i(t)}{dt} \\
&= -\sum_{i=1}^{n} C_i \frac{du_i(t)}{dt} \frac{dx_i(t)}{dt} = -\sum_{i=1}^{n} C_i h''(x_i(t)) \left(\frac{dx_i(t)}{dt} \right)^2 \leq 0,
\end{aligned} \tag{6.45}$$

因此，沿着解曲线，函数值 $\hat{E}(x(t))$ 非增加，$t \to \infty$ 的时候，可以期待 $x(t)$ 接近问题 (6.43) 的局部最优解. 另外，由于微分方程组 (6.44) 可在电路上实现，所以有可能用这种模拟计算模型求解问题 (6.43).

本节所考察的方法可适用于各种组合优化问题. 为此，有必要把要解的组合优化问题表示成问题 (6.32) 的形状，但是这方法随着问题的不同而不同，很难做一个统一的说明. 这里把第 3 章所考察过的**旅行商问题** (traveling salesman problem) 作为具体的例子举出，说明如何能把它模型化为问题 (6.32) 的形状. 这个问题是，在有 N 个节点 $v_1, v_2, \ldots, v_N$ 的无向完全图里，节点 v_i 和节点 v_j 的距离为 $c(v_i, v_j)$ 时，找出正好通过各节点一次的回路 (哈密尔顿回路) 中的最短者. 另外，在此假定对所有的 i, j 有 $c(v_i, v_j) = c(v_j, v_i)$ 以及 $c(v_i, v_i) = 0$. 如下所示，该问题

可用 N^2 个 0-1 变量 $x_{i\ell}$ $(i=1,2,\ldots,N,\ell=1,2,\ldots,N)$ 来模型化，其中变量 $x_{i\ell}$ 表示

$$x_{i\ell}=\begin{cases}1, & \text{节点 } v_i \text{ 在回路中出现在第 } \ell \text{ 号时}\\ 0, & \text{否则}\end{cases} \tag{6.46}$$

这时，$x=(x_{i\ell})$ 为了能表示哈密尔顿回路必须满足

$$\sum_{\ell=1}^{N} x_{i\ell}=1 \quad (i=1,2,\ldots,N) \tag{6.47}$$

$$\sum_{i=1}^{N} x_{i\ell}=1 \quad (\ell=1,2,\ldots,N) \tag{6.48}$$

另外，对应 $x=(x_{i\ell})$ 的回路长度由

$$F(x)=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}c(v_i,v_j)\left(\sum_{\ell=1}^{N}x_{i\ell}\left(x_{j,\ell-1}+x_{j,\ell+1}\right)\right) \tag{6.49}$$

给出．这里，式 (6.49) 中，当 $\ell=1$ 时约定 $\ell-1=N$，当 $\ell=N$ 时约定 $\ell+1=1$．于是，旅行商问题可表示成为，在约束条件 (6.47)，(6.48) 下，确定由式 (6.49) 所定义的函数 F 达最小的 0-1 变量向量 x 的问题．该问题包含有等式约束条件，因不具备问题 (6.32) 的形状，导入针对约束条件 (6.47)，(6.48) 的罚函数

$$G(x)=\sum_{i=1}^{N}\left(\sum_{\ell=1}^{N}x_{i\ell}-1\right)^2+\sum_{\ell=1}^{N}\left(\sum_{i=1}^{N}x_{i\ell}-1\right)^2 \tag{6.50}$$

后，变换成具备问题 (6.32) 的形状的问题

$$\begin{aligned}&\text{目标函数：}\quad F(x)+\rho G(x)\quad\longrightarrow\quad\text{最小}\\&\text{约束条件：}\quad x_i=0,1\ (i=1,2,\ldots,n)\end{aligned} \tag{6.51}$$

这里，罚参数 ρ 为足够大的正数．当然该问题有多个局部最优解，但是，如果解上述问题所得到的解实际上满足约束条件 (6.47), (6.48), 可以期望该解是旅行商问题的一个好的近似解．

最后确认一下问题 (6.51) 满足问题 (6.32) 的假设 (6.33), (6.34)．首先容易验证，函数 F 的海赛矩阵为对称的，而且对角元为 0．另外，关于函数 G, 如果照搬定义式 (6.50), 则海赛矩阵的对角元 ($x_{i\ell}^2$ 的系数) 不为 0, 但是，从 $x_{i\ell}$ 为 0-1 变量的条件出发，用 $x_{i\ell}$ 代替 $x_{i\ell}^2$ 的话，函数 G 可以表示为不包含项 $x_{i\ell}^2$ 的形式．再者，显然函数 G 的海赛矩阵对称，因此，可以明白问题 (6.51) 的目标函数的二次系数矩阵满足式 (6.33), (6.34) 的性质．

6.5 文献及其它话题

神经网络是近年来研究非常活跃的领域，出版了很多包括各种应用在内的论文及书．这里列举麻生 [A88], 中野 [N90], Rumelhart, McClelland 等 [RM86] 著作．另外，Anderson, Rosenfeld [AR88] 收集编辑了该领域的基础论文，对于想学习原著的读者来说很方便．

6.2 节的感知器可以说是 30 年以前提出的神经网络的雏型， Minsky, Papert [MP88] 考察了它的功能． 6.3 节和 6.4 节提到的误差逆传播法和组合优化法分别是 Rumelhart, Hinton, Williams [RHW86] 和 Hopfield [H82] 提出的想法，已经成为现在神经网络研究的先驱．另外，关于 6.4 节所涉及到的波尔兹曼机和模拟退火法，请参照 Ackley, Hinton, Sejnowski [AHS85] 和 Kirkpatrick, Gelatt, Vecchi [KGV83].

这一章从本书书名 — 最优化的观点讲解了有代表性的神

经网络. 特别是, 把作为多层网络的学习算法的误差逆传播法, 看成是对非线性联立方程组的逐次投影法的思想等, 在以往的神经网络教科书上没怎么提及. 关于逐次投影法的参考文献, 6.3 节所涉及到的针对线性不等式组的方法有 Censor [C81], 关于 6.4 节所提起的针对一般非线性方程式组的方法的收敛结果有 McCormick [M77].

附 录

A.1 线性代数

以 R 表示全体实数的集合，R^n 表示以实数为分量的全体 n 维向量的集合 (n 维实欧几里得空间) 空间 R^n 的元素也称为点. 以下把向量和点作为同义语使用，根据情况适当分开.

n 维向量 $x \in R^n$ 的分量用下标写成 $x_1, x_2, \ldots, x_n$ 或者 x_i ($i = 1, 2, \ldots, n$). 与此同时，向量列或者数列用 (带括弧的) 上标表示成为 $x^{(1)}, x^{(2)}, \ldots$, 或者 $\{x^{(k)}\}$

本书在不特别声明的情况下，向量是列向量，向量 x 的转置用 x^T 表示. 对 n 维向量 x, $\sqrt{x^T x} = \sqrt{\sum_{i=1}^n x_i^2}$ 所给出的量称为 x 的**欧几里得范数** (**Euclidean norm**), 写成 $\|x\|$. 基于这个范数，空间 R^n 的两点 x、y 的**距离** (**distance**) 用 $\|x - y\|$ 确定.

对空间 R^n 的点列 $\{x^{(k)}\}$, 存在满足 $\lim_{k\to\infty} \|x^{(k)} - x^*\| = 0$ 的点 $x^* \in R^n$ 的时候，称点 x^* 为点列 $\{x^{(k)}\}$ 的**极限** (limit), 点列 $\{x^{(k)}\}$ **收敛** (converge) 于点 x^*. 另外，即使在点列 $\{x^{(k)}\}$ 作为整体不收敛于一个极限的情况下，只要能够适当选取其子列 $\{x^{(k_i)}\}$ 使得 $\{x^{(k_i)}\}$ 收敛于点 x^* 的时候，称 x^* 为点列 $\{x^{(k)}\}$ 的**聚点** (accumulation point). 对点列 $\{x^{(k)}\}$ 存在满足 $\|x^{(k)}\| < M (k = 1, 2, \ldots)$ 的正常数 M 的时候，称 $\{x^{(k)}\}$ **有界** (bounded). 这样的点列至少具有一个聚点.

对空间 R^n 的子集合 X, 连接 X 内任意两点的线段也包含

在 X 内，即成立

$$x, y \in X, 0 \leq \alpha \leq 1 \implies (1-\alpha)x + \alpha y \in X$$

的时候[1), 称 X 为**凸集合** (convex set) (参照图 A.1). 任意个数的凸集合的共同部分也是凸集合. 对空间 R^n 的任意子集合 X, 称包含它的最小的凸集合称为 X 的**凸包** (convex hull), 用 $\operatorname{conv} X$ 表示.

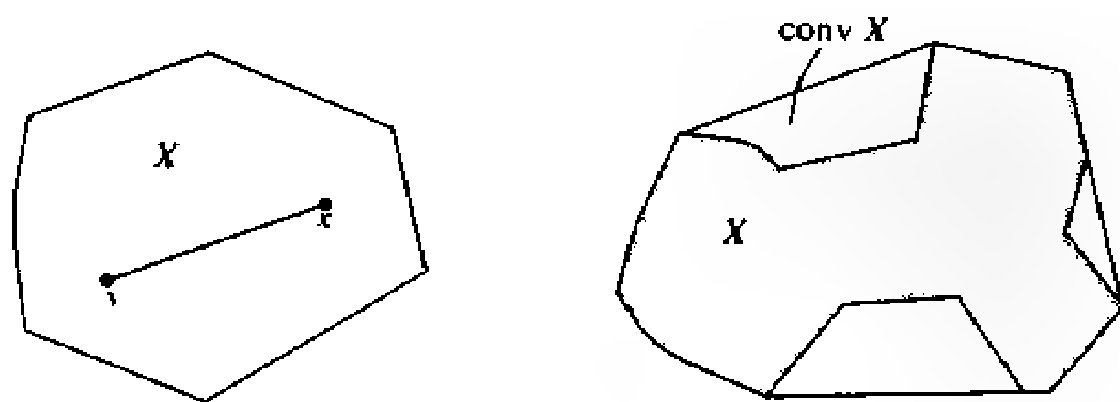


图 A.1 凸集合和凸包

对以实数为分量的 m 行 n 列矩阵 ($m \times n$ 矩阵) A, 如果它的 (i,j) 分量为 a_{ij}, 则把该矩阵简单记为 $A = (a_{ij})$. $n \times n$ 矩阵 A 的分量为 $a_{ii} = 1(i = 1, 2, \ldots, n)$ 而且满足 $a_{ij} = 0(i, j = 1, 2, \ldots, n, i \neq j)$ 的时候, 称 A 为**单位矩阵** (unit matrix), 用 I 表示. 对于 $n \times n$ 矩阵 A 如果存在满足 $AB = I$ 的 $n \times n$ 矩阵 B, 则称 A 为**非奇异矩阵** (nonsingular matrix). 另外, 这个时候称矩阵 B 为 A 的**逆矩阵** (inverse matrix), 用 A^{-1} 表示.

对于 $n \times n$ 矩阵 A, 分别称满足 $Ax = \lambda x$ 的数 λ 和 n 维向量 $x \in R^n$ 为 A 的**特征值** (eigenvalue) 和**特征向量** (eigenvector). 另外, 如果成立 $a_{ij} = a_{ji}$ $(i, j = 1, 2, \ldots, n)$, 则称 A 为**对称矩阵**

1) 这里, "$A \Longrightarrow B$" 表示"如果 A 则 B"的命题.

(symmetric matrix). 对称矩阵特征值全为实数. 对于 $n\times n$ 对称矩阵 A 在成立

$$x^TAx>0\quad(\text{对任意}x\in R^n,\ x\neq 0)$$

的时候，称 A 为**正定矩阵** (positive definite matrix), 在成立

$$x^TAx\geq 0\quad(\text{对任意}x\in R^n)$$

的时候，称 A 为**半正定矩阵** (positive semi-definite matrix)[1]. 正定矩阵的特征值全为正，半正定矩阵的特征值全非负.

在 $n\times n$ 矩阵 $L=(l_{ij})$ 里，对所有的 $i<j$ 都有 $l_{ij}=0$ 的时候，称 L 为**下三角矩阵** (lower triangular matrix), 把转置矩阵为下三角矩阵的矩阵称为**上三角矩阵** (upper triangular matrix). 把 $n\times n$ 矩阵 A 表示为下三角矩阵 L 和上三角矩阵 U 的积 $A=LU$ 的形式称为 **LU 分解** (LU decomposition). 在得到矩阵 A 的 LU 分解 $A=LU$ 的时候，联立 1 次方程式 $Ax=b$ 可以表达成为 $LUx=b$, 故可以通过先求出方程式 $Ly=b$ 的解 y, 再解方程式 $Ux=y$ 求出 x 的两阶段来计算解 x. 进一步，两方程式 $Ly=b$, $Ux=y$ 可以分别仅仅采用被称为前进代入和后退代入的代入操作来解. 特别是，如果矩阵 A 为正定对称，则利用下三角矩阵 L 可以分解为 $A=LL^T$. 称这是 **Cholesky 分解** (Cholesky decomposition).

A.2 多变量函数

从集合 X 到集合 Y 的映射 f 记为 $f:X\to Y$. 特别地，映射 $f:R^n\to R$ 表示有 n 个实变量的实数值函数. 对函数 $f:R^n\to R$ 成立

1) 有时也把半正定矩阵称为非负定矩阵.

$$x, y \in X, 0 \leq \alpha \leq 1 \Longrightarrow f((1-\alpha)x + \alpha y) < (1-\alpha)f(x) + \alpha f(y)$$

的时候，称 f 为**凸函数** (convex function) (图 A.2).

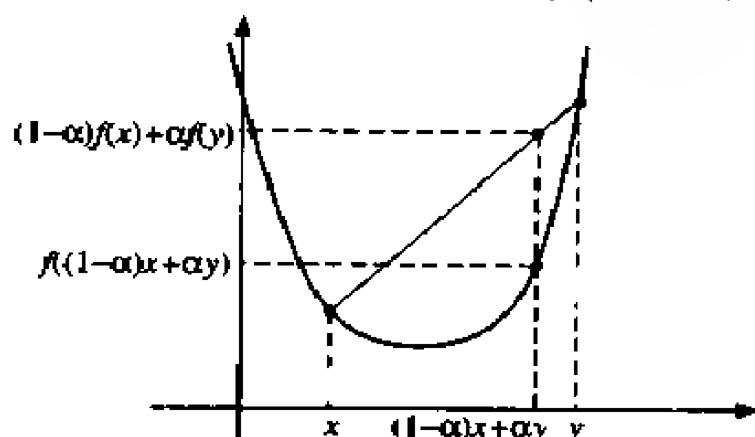


图 A.2 单变量的凸函数例

函数 $f: R^n \to R$ 中固定变量 x_i 以外的 $n-1$ 个变量，把 f 看成仅含变量 x_i 的单变量函数，如果它可微，则称 f 关于 x_i**偏可微** (partially differentiable). 另外其微分系数称为函数 f 关于变量 x_i 的**偏微分系数** (partial differential coefficient), 写成 $\dfrac{\partial f(x)}{\partial x_i}$. 进一步，把 $\dfrac{\partial f(x)}{\partial x_i}$ 看成 x 的函数，称之为 f 关于 x_i 的**偏导函数** (partial derivative).

如函数 $f: R^n \to R$ 的关于 x_i 的偏导函数 $\dfrac{\partial f(x)}{\partial x_i}$ 进一步关于变量 x_j 偏可微，就可得到二次偏微分系数 $\dfrac{\partial^2 f(x)}{\partial x_j \partial x_i} = \dfrac{\partial}{\partial x_j}\left(\dfrac{\partial f(x)}{\partial x_i}\right)$ 以及二次偏导函数.

称以函数 $f: R^n \to R$ 关于各变量 x_i 的偏微分系数为分量的 n 维向量

$$\left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \cdots, \frac{\partial f(x)}{\partial x_n}\right)^T$$

为 f 的**梯度** (gradient), 用 $\nabla f(x)$ 表示. 进一步，称以二次偏微分系数 $\dfrac{\partial^2 f(x)}{\partial x_i \partial x_j}$ 为第 (i,j) 分量的 $n \times n$ 矩阵为 f 的**海赛矩阵**

(Hessian matrix), 用 $\nabla^2 f(x)$ 表示. 如果所有的二次偏导函数连续, 则海赛矩阵 $\nabla^2 f(x)$ 对称. 进一步, 如果它在任意的 x 处半正定则 f 是凸函数. 另外, 根据针对多变量函数的 **泰勒定理** (Taylor's theorem), 对任意向量 $x \in R^n$ 和 $d \in R^n$, 存在满足

$$\begin{aligned} f(x+d) &= f(x)+\sum_{i=1}^{n}\frac{\partial f(x)}{\partial x_i}d_i+\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2 f(x+\theta d)}{\partial x_i \partial x_j}d_i d_j \\ &= f(x)+\nabla f(x)^T d+\frac{1}{2}d^T\nabla^2 f(x+\theta d)d \end{aligned}$$

的 $\theta \in (0,1)$, 进一步成立

$$f(x+d)=f(x)+\nabla f(x)^T d+\frac{1}{2}d^T\nabla^2 f(x)d+o(||d||^2).$$

这里, $o: R \to R$ 是 $t \to 0$ 的时候成立 $o(t) \to 0$ 的适当函数.

A.3 图论

称有限个 **节点** (node, vertex) 的集合 $V=\{v_1, v_2, \dots, v_n\}$ 和节点对的集合 $E \subseteq V \times V \equiv \{(v_i, v_j) \mid v_i \in V, v_j \in V\}$ 构成的组为 **图** (graph), 用 $G=(V,E)$ 表示. 称属于集合 E 的节点对 $e=(v,w)$ 为图 G 的 **边** (arc, branch, edge), 称节点 v 和 w 为边 e 的 **端点** (end nodes), 边 e 连接节点 v, w. 不考虑图 G 里各边的方向时称为 **无向图** (undirected graph), 考虑方向因而区别边 (v,w) 和 (w,v) 的时候称为 **有向图** (directed graph) (图 A.3 (a) 和 (b)). 有向图里的边 $e=(v,w)$ 也被称为 **有向边** (directed arc), 分别称节点 v 和 w 为边 (v,w) 的 **始点** (tail) 和 **终点** (head). 对节点 v, 称连接它的边的数目为 v 的 **度数** (degree), 特别是, 称有向图里以节点 v 为始点的边数为 v 的 **出度数** (out-degree), 以节点 v 为终点的边数为 v 的 **入度数** (in-degree). 另外, 在有向图的情况

下，把以节点 v 为始点的边集合记为 $OUT(v)$，以节点 v 为终点的边集合记为 $IN(v)$.

具有 n 个节点的无向图 G 里所有节点间存在有边的时候，称 G 为**完全图** (complete graph)，记为 K_n (图 A.3 (c)).

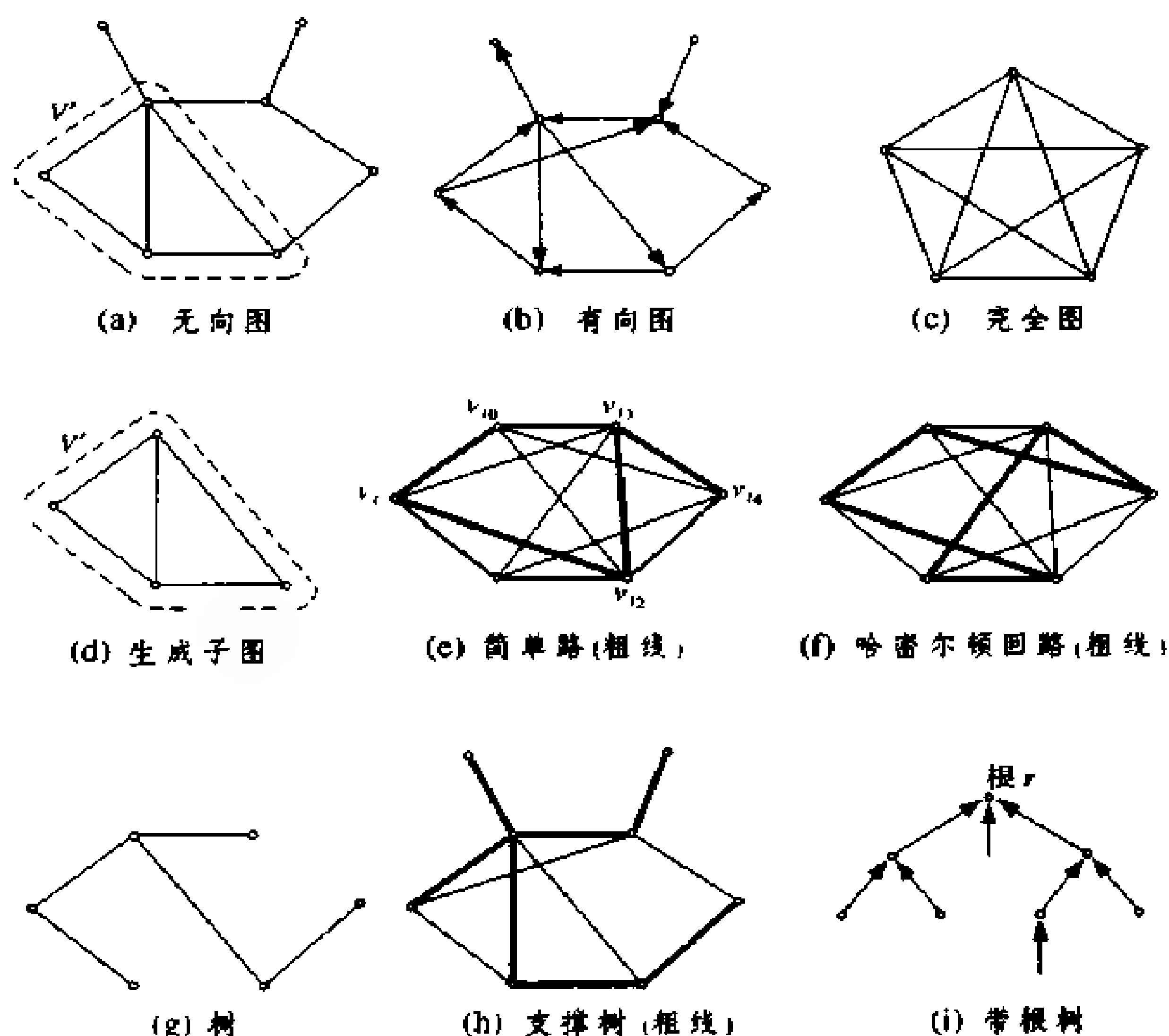


图 A.3 图

对图 $G=(V,E)$，考虑子集合 $V'\subseteq V$ 和 $E'\subseteq E$，如果任意的 $e'\in E'$ 的两端点属于 V'，则 $G'=(V',E')$ 也成为图. 称这种 G' 为 G 的**子图** (subgraph). 特别是，图 $G=(V,E)$ 里，两端点在

集合 $V' \subseteq V$ 的边集合与 $E' \subseteq E$ 一致的时候，称 $G' = (V', E')$ 为由 V' 产生的 G 的 **生成子图** (induced subgraph) (图 A.3 (d) 是 (a) 的生成子图).

图 $G = (V, E)$ 的节点列 $\pi = v_{i_0} v_{i_1} \cdots v_{i_k}$ 满足 $(v_{i_j}, v_{i_{j+1}}) \in E$ $(j = 0, 1, \ldots, k-1)$ 的时候，称 π 为从节点 v_{i_0} 到 v_{i_k} 的 **路** (path). 路 π 可以看成构成它的节点的集合或者边的集合，有时采用 $v_{i_j} \in \pi$ 以及 $(v_{i_j}, v_{i_{j+1}}) \in \pi$ 等记法. 另外，这个时候称从 v_{i_0} 可达 (reachable) v_{i_k}. 进一步，路 π 的始点 v_{i_0} 和终点 v_{i_k} 相同的时候称之为 **回路** (cycle, circuit), 包含在路里的节点 $v_{i_0}, v_{i_1}, \ldots, v_{i_{k-1}}$ 全不同的时候称之为 **简单路** (simple path) (图 A.3 (e)), 同时具有两个性质的时候称之为 **简单回路** (simple cycle). 称通过图 G 的所有节点的简单回路为 **哈密尔顿回路** (Hamiltonian cycle) (图 A.3 (f)).

无向图 G (忽略有向图各边的方向可以得到无向图) 里，任意两个节点间存在有路的时候，称 G 为 **连通图** (connected graph). 对一个并不一定连通的图 G, 在它的连通生成子图中，称不存在真包含在其中的连通生成子图者为 G 的连通分支. 不连通的图可以分割成为相互不共节点的几个连通分支. 连通无向图 G 不包含回路的时候，称 G 为 **树** (tree) (图 A.3 (g)). 具有 n 节点的连通无向图 G 成为树的充分必要条件是 G 刚好具有 $n-1$ 条边. 对无向图 $G = (V, E)$, 具有与 G 同样的节点集合的子图 $G' = (V, E')$ 为树的时候，称 G' 为 G 的 **生成树** (spanning tree) (图 A.3 (h)).

以下考虑有向图的树. 把树中存在有被称为 **根** (root) 的一个特别节点 r 者称为 **带根树** (rooted tree) (图 A.3 (i)). 带根树里存在有从任意节点 v 到 r 的有向路. 这个从 v 到根 r 的路是唯一确定的，称其中在节点 v 后出现的节点为 v 的 **父亲** (parent),

节点 w 为 v 的父亲的时候称 v 为 w 的 **孩子** (child). 另外，把节点 v 到根 r 的路上存在的任意的节点记为 u, 称 u 为 v 的 **祖先** (ancestor), v 为 u 的 **子孙** (descendant). 另外，把除自己以外无其它子孙的节点称为 **叶** (leaf). 对带根树的各节点 v, 称从 v 到根的路的长度 (所包含的边数) 为 v 的 **深度** (depth). 所有成为叶的 v 的子孙到 v 的路之中，称其最长者的长度称为 v 的 **高度** (height), 根自身的高度称为 **树的高度** (tree height). 以带根树的任一个节点为根，称它的所有子孙形成的树为 **子树** (subtree).

A.4 计算复杂性

一个 **问题** (problem) 通常由无限个 **问题实例** (problem instances) 组成. 比如，旅行商问题是通过具体给定节点数和各节点对间的距离后确定的无数问题实例的集合. 所谓设计解某个问题的算法，是在给定属于该问题的任何实例的情况下，都能正确找到它的解.

算法一般由加减乘除和大小比较等基本步骤组成，称找出某个问题的解所需要的这些基本步骤的执行次数称为 **时间复杂性** (time complexity) (也称时间量). 相应地，称为了保持计算中间结果所必要的存储空间大小为 **空间复杂性** (space complexity). 另外，把它们综合起来称为 **计算复杂性** (complexity), 也称计算量.

另外，处理数字的算法之基本步骤严格说来有如下两种取法. 其一是把加减乘除等运算都各自算作一步，其二是把这些运算分解成二进制数的位数 (bit) 运算形式之后对位数运算加以计数. 称前者为 **单位成本** (unit cost) **法**, 后者为 **对数成本** (logarithmic cost) **法**. 后者的取名是因为自然数 x 化为二进制时

其位数大致为 $\log_2 x$. 采用该方法时需要的成本因所处理的数 x 之大小不同而不一样. 例如, 在 n 位数的两个数求和的情况下, 采用单位成本法时其计算量是 1, 而采用对数成本法时, 其计算量被估为 $c_1 n + c_2$ (这里, c_1 和 c_2 为特定的常数).

当所有的数通过一定的位数 (比如计算机中一个词的位数) 能够以足够的精度表示出来时, 采用单位成本法比较合适; 当不同的数字需要不同位数才能无误差地表示出来时采用对数成本法比较合适. 在大多数情况下采用单位成本法对本书的算法进行评估比较现实 (但是, 对第 5 章的椭球法和内点法评估时采用了对数成本法).

算法的计算量因问题实例不同而不同, 一般说来, 如果是大规模问题实例则计算量也相应地大起来. 于是, 对算法进行评价时, 有必要调查随着问题实例的规模的增大计算量如何变化. 作为表达问题实例规模的指标, 通常采用输入它的数据长 (位数). 比如, 旅行商问题里, 如果节点数为 n, 节点对之间距离的最大值为 C, 则输入数据长可以用 $O(n^2 \log C)$ 来评价 (对数成本法)[1]. 这里, $O(f(N))$ 读作 $f(N)$ 阶, $T(N) = O(f(N))$ 表示存在某个正常数 c 和 N_0, 当 $N \geq N_0$ 时成立 $T(N) \leq cf(N)$. 比如, $0.5N^2, 100N + 5N^2, 50 - 10^4 N + 10^{-3} N^2$ 等全可写为 $O(N^2)$. 特别是, 当 $f(N)$ 是不依赖于 N 的常数时, 把 $O(f(N))$ 写成 $O(1)$.

当给定规模为 $O(N)$ 的问题实例时, 想把针对该问题的算法计算量的阶用 N 的函数来表示. 但是, 规模为 $O(N)$ 的问题实例一般存在有无数个, 故有必要用某种客观的基准来进行整体评价. 为此, 一般采用如下两个基准之一:

最坏计算量 (worst case complexity): 基于规模为 N 的所有

1) 设节点对之间的距离全为整数.

问题实例中需要最大计算量者来确定.

平均计算量 (average complexity): 基于规模为 N 的问题实例各自发生的概率来确定.

有时有一部分问题实例需要异常多的计算量, 前者会因此给出过于悲观的评价. 然而, 与后者比起来, 前者在数学处理上要容易些, 因而被经常采用.

在评价这些算法效率的时候, 经常考虑的问题是时间量 $O(f(N))$ 是不是**多项式阶** (polynomial order). 所谓多项式阶是说可以用某个常数 k 写为 $O(N^k)$. 但是也有像 $O(N^2 \log N)(\leq O(N^3))$ 这样含有 $\log N$ 的情况. 与此相反, 不是多项式阶的例有 $O(N^{\log N})$, $O(2^N)$, $O(N!)$ 等等.

称时间量是多项式阶的算法为**多项式 (时间) 算法** (polynomial time algorithm). 针对一个问题一般可以认为有多个算法, 当知道至少存在有一个多项式时间算法时, 称该问题属于 **P 类**. 实用上可以认为属于 P 类的问题是可解的问题. 本书所讨论的问题中, 线性规划问题, 2 次规划问题, 最大流问题, 最小费用流问题等属于 P 类. 与此相反, 有很多像旅行商问题那样不知道存在有多项式时间算法的问题. **NP 困难性** (NP-hardness) 的概念被认为是给出不存在多项式时间算法的理论根据, 比如旅行商问题也是 NP 困难的.

文　献

（第 1 章）

[B90] J.E Beasley: Linear programming on Cray supercomputers, *Journal of Operational Research Society*, Vol 41 (1990), pp. 133–139.

[BGLMS92] R.E. Bixby, J.W. Gregory, I.J. Lustig, R.E. Marsten, D.F. Shanno: Very large-scale programming: A case study in combining interior point and simplex methods, *Operations Research*, Vol. 40 (1992), pp. 885–897.

[B77] R.G. Bland New finite pivoting rules for the simplex method, *Mathematics of Operations Research*, Vol. 2 (1977), pp. 103–107

[C83] V. Chvátal: *Linear Programming*, W H. Freeman and Company, 1983. 日文版 (阪田省二郎等译), 线性规划法 (上 · 下), 启学出版, 1986/1988.

[D63] G.B. Dantzig: *Linear Programming and Extensions*, Princeton University Press, 1963. 日文版 (小山昭雄译) 线性规划法及其扩张, ホルト・サウンダース, 1983.

[GT89] D. Goldfarb, M.J Todd. Linear programming, *Handbooks in Operations Research and Management Science, Volume 1, Optimization*, (Eds. G.L. Nemhauser et al.), North-Holland, pp. 73–170, 1989.

[I86] 伊理正夫：线性规划法，共立出版，1986.

[IF91] 茨木俊秀，福岛雅夫：FORTRAN 77 最优化程序，岩波书店，1991.

[K80] 古林隆：线性规划法入门，产业图书，1980.

[K87] 今野浩：线性规划法，日科技连出版社，1987.

[S87] R. Shamir: The efficiency of the simplex method: A survey, *Management Science*, Vol. 33 (1987), pp. 301–334.

(第 2 章)

[AMO 93] R. K. Ahuja, T L. Magnanti, J. B. Orlin: *Network Flows: Theory, Algorithms and Application*, Prentice Hall, 1993.

[D 59] E. W. Dijkstra: A note on two problems in connexion with graphs, *Numerische Mathematik*, Vol. 1 (1959), pp. 269–271

[EK 72] J. Edmonds, R. M. Karp: Theoretical improvements in algorithmic efficiency of network flow problem, *J. of ACM*, Vol. 19 (1972), pp.248–264.

[FF 62] L. R. Ford, D. R. Fulkerson: *Flows in Networks*, Princeton University Press, 1962.

[FT 87] M. L. Fredman, R. E. Tarjan: Fibonacci heaps and their uses in improved network optimization algorithms, *J. of ACM*, Vol. 34 (1987), pp. 596–615.

[F 86] S. Fujishige: A capacity-rounding algorithm for the minimum-cost circulation problem: A dual framework of the Tardos algorithm, *Mathematical Programming*, Vol. 35 (1986), pp. 298–308.

[GT 88] A. V Goldberg, R. E. Tarjan: A new approach to the maximum flow problem, *J. of ACM*, Vol. 35 (1988), pp. 921–940

[GT 89] A. V. Goldberg, R. E Tarjan: Finding minimum-cost circulations by canceling negative cycles, *J. of ACM*, Vol. 36 (1989), pp. 873–886

[IF 91] 茨木俊秀，福岛雅夫： FORTRAN 77 最优化程序，岩波书店， 1991.

[II 84] H. Imai, M. Iri Practical efficiencies of existing shortest path algorithms and a new bucket algorithm, *J. Operations Research Society of Japan*, Vol. 27 (1984), pp. 43–85.

[IFO 86] 伊理正夫、藤重悟，大山达雄：图 · 网络 · 拟阵，产业图书， 1986.

[I 89] 岩野和生：最大流算法最近的发展及其背景 — I 和 II, 情报处理， Vol. 30 (1989), pp. 1494 1501 和 Vol. 31 (1990), pp. 82 88.

[K 78] R. M. Karp: A characterization of the minimum cycle mean in a graph, *Discrete Mathematics*, Vol. 23 (1978), pp. 309–311.

[N 90] 永持仁：组合优化里 Tardos 的新解法，系统 / 控制 / 信息， Vol. 34 (1990), pp 232 240.

[O 88] J. B Orlin: A faster strongly polynomial minimum cost flow algorithm, *20 ACM Symposium on Theory of Computing (STOC)*, pp 377 387, 1988.

[T 85] E. Tardos: A strongly polynomial minimum cost circulation algorithm, *Combinatorica*, Vol. 5 (1985), pp. 247 255

[T 83] R. E. Tarjan: *Data Structures and Network Algorithms*, SIAM Publications 1983. 日文版 (岩野和生译), 数据结构和网络算法, マグロウヒル, 1989.

(第 3 章)

[BC 91] S. C. Boyd, W. H. Cunningham: Small traveling salesman polytipes, *Mathematics of Operations Research*, Vol 16 (1991), pp 259 271

[CFN 85] G. Cornuéjols, J. Fonlupt, D. Naddef: The traveling salesman problem on a graph and some related integer polyhedra, *Mathematical Programming*, Vol. 33 (1985), pp. 1 27.

[DFJ 54] G B Dantizig, D. R. Fulkerson, S. M. Johnson: Solution of a large scale traveling–salesman problem, *Operations Research*, Vol. 2 (1954), pp. 393–410.

[GH 61] R. E. Gomory, T. C. Hu Multi-terminal network flows, *SIAM J. on Applied Mathematics*, Vol 9 (1961), pp. 551–570.

[GH 91] M. Grötschel, O. Holland: Solution of large-scale symmetric traveling salesman problems, *Mathematical Programming*, Vol. 51 (1991), pp 141 202

[GP 79] M. Grotschel, M W. Padberg: On the symmetric traveling salesman problem I (inequalities) and II (lifting theorems and facets), *Mathematical Programming*, Vol. 16 (1979), pp. 265–280 and pp. 281 302.

[GP 86] M. Grötschel, W. R. Pulleyblank: Clique tree inequalities and the symmetric traveling salesman problem, *Mathematics of Operations Research*, Vol. 11 (1986), pp. 537-569.

[I 83] 茨木俊秀：组合优化 — 以分枝定界法为中心，产业图书， 1983

[LLRS 85] E. L. Lawler, J K Lenstra, A. H. G Rinnooy Kan, D. B Shmoys (eds.): *The Traveling Salesman Problem*, Wiley, 1985.

[LK 73] S. Lin, B. W. Kernighan: An efficient heuristic algorithm for the traveling salesman problem, *Operations Research*, Vol. 21 (1973), pp. 498–516.

[NR 91] D. Naddef, G. Rinaldi: The symmetric traveling salesman polytope and its graphical relaxation: Composition of valid inequalities, *Mathematical Programming*, Vol. 51 (1991), pp. 359 400

[NI 92] H. Nagamochi, T. Ibaraki: Computing edge-connectivity in multigraphs and capacitated graphs, *SIAM J on Discrete Mathematics*, Vol. 5 (1992), pp. 54 66

[PR 91] M. W. Padberg, G. Rinaldi: An efficient algorithm for the minimum capacity cut problem, *Mathematical Programming*, Vol. 47 (1990), pp. 19–36.

[PR 91] M. W. Padberg, G. Rinaldi: A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, *SIAM Review*, Vol. 33 (1991), pp. 60-100.

(第 4 章)

[A91] ASNOP 研究会 (编)：个人计算机 FORTRAN 版非线性最优化程序，日刊工业新闻社， 1991.

[B82] D.P. Bertsekas: *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, 1982.

[BT89] D.P. Bertsekas, J.N. Tsitsiklis: *Parallel and Distributed Computation – Numerical Methods*, Prentice-Hall, 1989.

[DS83] J.E. Dennis, Jr , R.B. Schnabel: *Numerical Methods for Unconstrained optimization and Nonlinear Equations*, Prentice-Hall, 1983

[DS89] J.E. Dennis, Jr., R.B. Schnabel. A view of unconstrained optimization, *Handbooks in Operations Research and Management Science, Volume 1, Optimization*, (Eds. G.L. Nemhauser et al.), North-Holland, pp. 1–72, 1989.

[F87] R. Fletcher: *Practical Methods of Optimization: Second Edition*, John Wiley & Sons, 1987.

[FNT91] 布川昊，中山弘隆，谷野哲三：线性代数和凸解析，コロナ社， 1991.

[F80] 福岛雅夫：非线性最优化的理论，产业图书， 1980.

[GMW81] P.E. Gill, W. Murray, M.H. Wright *Practical Optimization*, Academic Press, 1981.

[GMSW89] P.E. Gill, W Murray, M.A. Saunders, M.H. Wright: Constrained nonlinear programming, *Handbooks in Operations Research and Management Science, Volume 1, Optimization*, (Eds. G.L. Nemhauser et al.), North-Holland, pp. 171–210, 1989.

[HP90] P.T Harker, J.-S. Pang: Finite-dimensional variational inequality and nonlinear complementarity problems. A survey of theory, algorithms and applications, *Mathematical Programming*, Vol. 48 (1990), pp. 161 220.

[IF91] 茨木俊秀，福島雅夫： FORTRAN77 最优化程序，岩波书店， 1991.

[KY78] 今野浩，山下浩：非线性规划法，日科技连出版社，1978.

[R70] R.T. Rockafellar: *Convex Analysis*, Princeton University Press, 1970.

[R76] R.T. Rockafellar Monotone operators and the proximal point algorithm, *SIAM Journal on Control and Optimization*, Vol. 14 (1976), pp. 877 – 898.

(第 5 章)

[AKRV89] I. Adler, N. Karmarkar, M.C. Resende, G. Veiga: Data structures and programming techniques for the implementation

of Karmarkar's algorithm, *ORSA Journal on Computing*, Vol 1 (1989), pp 84–106.

[B86] E.R. Barnes: A variation of Karmarkar's algorithm for solving linear programming problems, *Mathematical Programming*, Vol 36 (1986), pp. 174 182.

[BGLMS92] R.E. Bixby, J.W. Gregory, I.J. Lustig, R.E. Marsten, D F. Shanno: Very large-scale programming: A case study in combining interior point and simplex methods, *Operations Research*, Vol. 40 (1992), pp. 885 897.

[BGT81] R.G. Bland, D. Goldfarb, M.J. Todd: The ellipsoid method: A survey, *Operations Research*, Vol. 29 (1981), pp. 1039–1091

[C83] V. Chvátal: *Linear Programming*, W.H. Freeman and Company, 1983. 日文版 (阪田省二郎等译) 线性规划法 (上 · 下), 启学出版, 1986/1988.

[D67] I.I. Dikin: Iterative solution of problems of linear and quadratic programming, *Soviet Mathematics Doklady*, Vol. 8 (1967), pp. 674 675

[GT89] D Goldfarb, M.J Todd: Linear programming, *Handbooks in Operations Research and Management Science, Volume 1, Optimization*, (Eds. G.L. Nemhauser et al), North-Holland, pp. 73 170, 1989

[I86] 伊理正夫: 线性规划法, 共立出版, 1986.

[K84] N. Karmarkar: A new polynomial time algorithm for linear programming, *Combinatorica*, Vol. 4 (1984), pp. 373–395.

[K79] L.G. Khachiyan: A polynomial algorithm in linear programming, *Soviet Mathematics Doklady*, Vol. 20 (1979), pp. 191–194.

[KMNY91] M. Kojima, N. Megiddo, T. Noma, A. Yoshise. *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems*, Spinger-Verlag, 1991.

[LMS92] I.J. Lustig, R.E. Marsten, D.F. Shanno: On implementing Mehrotra's predictor-corrector interior-point method for linear programming, *SIAM Journal on Optimization*, Vol. 2 (1992), pp. 435–449.

[K87] 今野浩：线性规划法，日科技连出版社， 1987.

[M92] S. Mehrotra. Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods, *ORSA Journal on Computing*, Vol. 4 (1992), pp. 103–118.

[MMS89] K. McShane, C.L. Monma, D. Shanno. An implementation of a primal-dual interior point method for linear programming, *ORSA Journal on Computing*, Vol. 1 (1989), pp. 70–83.

[MM87] C.L. Monma, A.J. Morton: Computational experience with a dual variant of Karmarkar's method for linear programming, *Operations Research Letters*, Vol. 6 (1987), pp. 261–267.

[T91] T. Tsuchiya: Global convergence of the affine scaling methods for degenerate linear programming problems, *Mathematical Programming*, Vol. 52 (1991), pp. 377–404.

[Y91] Y Ye· An $O(n^3L)$ potential reduction algorithm for linear programming, *Mathematical Programming*, Vol. 50 (1991), pp. 239–258.

(第 6 章)

[AHS85] D.H. Ackley, G.E. Hinton, T.J. Sejnowski: A learning algorithm for Boltzman machine, *Cognitive Science*, Vol. 9 (1985), pp. 147 169.

[AR88] J.A. Anderson, E. Rosenfeld (eds.): *Neurocomputing — Foundations of Research*, The MIT Press, 1988.

[A88] 麻生英树：神经网络情报处理，产业图书， 1988.

[C81] Y. Censor: Row-action methods for huge and sparse systems and their applications, *SIAM Review*, Vol 23 (1981), pp. 444–466.

[H82] J.J. Hopfield: Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, Vol. 79 (1982), pp. 2554–2558.

[KGV83] S. Kirkpatrick, C.D. Gelatt, Jr , M.P. Vecchi: Optimization by simulated annealing, *Science*, Vol 220 (1983), pp. 671–680.

[M77] S F. McCormick: The methods of Kaczmarz and row orthogonalization for solving linear equations and least squares problems in Hilbert space, *Indiana University Mathematical Journal*, Vol. 26 (1977), pp. 1137–1150.

[MP88] M. L. Minsky, S.A. Papert: *Perceptrons: Third Edition*, The MIT Press, 1988.

[N90] 中野馨：神经元计算机的基础，コロナ社， 1990.

[RHW86] D.E. Rumelhart, G.E. Hinton, R.J. Williams: Learning representations by back-propagating errors, *Nature*, Vol. 323 (1986), pp. 533–536.

[RM86] D.E. Rumelhart, J.L. McClelland, PDP Research Group *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volumes 1 and 2*, The MIT Press, 1986. 日文版 (甘利俊一监译) PDP 模型　认知科学和神经元回路网的探索，产业图书， 1989.

Index

【一画】

【二画】

【三画】

【四画】

【五画】

【六画】

【七画】

【九画】

【十画】

【十 画】

【十二画】

【十三画】

【十四画】

译者后记

一衣带水的邻国，中国和日本的文化交流历史源远流长．中国运筹学界的元老许国志先生和日本运筹学界的元老三根久先生在长年的学术交流中结下了深厚的友谊，为两国运筹学界的学术发展作出了非常重要的贡献．

本书的两位作者均是三根久先生的得意门生，现执教于日本一流大学 — 京都大学．茨木俊秀教授在离散型优化特别是组合优化算法方面、福岛雅夫教授在连续型优化特别是非线性规划方面造诣颇深，是世界知名的一流学者．除日本国内，茨木教授兼任着美国 University of Illinois, Rutgers University, 加拿大 Simon Fraser University 和 University of Waterloo 等大学的客座教授，福岛教授兼任着加拿大 University of Waterloo, 澳大利亚 University of New South Wales, 比利时 Faclutés Universitaires ND de la Paix 等大学的客座教授．

有幸获得在他们手下学习的机会，我非常钦佩他们学识渊博并深深受益于此．考虑到由于日语的障碍，很多中国的运筹学界的研究者难于了解邻国日本的发展状况，我一直想借当学生的机会把他们的研究成果介绍给国内同行．这个想法得到了来日本访问的中科院应用数学研究所所长章祥荪教授，曲阜师范大学的王长钰教授，山东师范大学的赵庆祯教授的大力支持．因离开中国多年，在翻译术语时碰到了很大的困难．为此，赵庆祯教授和章祥荪教授主动承担起繁杂的校正工作，改正了

初稿中的不少错误.（当然，遗留的错误当由译者本人负责.）初稿完成后，章祥荪教授又帮助联系世界图书出版公司，该出版公司的王炜女士也积极与日本共立出版社商谈版权事宜. 为加快出版速度，章祥荪教授的高徒，韦民和王磊两位同学奉献出宝贵的时间帮助我利用计算机排版. 多亏他们的努力，该书中文版才得以问世.

两位教授著作甚丰. 我选译本书是因为这是他们的最新著作. 本书在京都大学作为研究生教材使用，受到好评. 目前正准备再版.

谨以此译稿作为宝贵的京都大学留学生活纪念.

曾 道 智

1997 年 2 月于香川大学